

Cloud Computing

Gregor von Laszewski

Editor

laszewski@gmail.com

<https://cloudmesh-community.github.io/pub/vonlaszewski-draft.epub>

November 23, 2019 - 05:22 AM

Created by Cloudmesh & Cyberaide Bookmanager, <https://github.com/cyberaide/bookmanager>

CLOUD COMPUTING

Gregor von Laszewski

(c) Gregor von Laszewski, 2018, 2019

CLOUD COMPUTING

1 JULIA

1.1 Introduction to Julia for High Performance Computing 🌸

1.1.1 Introduction

1.1.1.1 Development Environments

1.1.2 Module Management

1.1.3 Multiple Dispatch

1.1.4 Parallel Language Constructs

1.1.5 Interfacing with the System

1.1.6 Installing Julia 🌸

1.1.6.1 Julia REPL

1.1.6.1.1 Linux

1.1.6.1.2 Windows

1.1.6.2 JuliaBox

1.1.7 Julia IDE 🌸

1.1.7.1 Install Atom

1.1.7.2 Install Juno

1.1.8 Packages and Modules 🌸

1.1.8.1 Installing Registered Modules

1.1.8.2 Custom Modules

1.2 Docopts 🌸

1.2.1 Installation

1.3 External programs 🌸

1.3.1 Resources

1.3.2 Python

1.3.2.1 Installation

1.3.2.2 Import Python Modules

1.3.3 Java in Julia

1.3.3.1 Installation

1.3.3.2 Usage

1.4 Parallel Computing 🌸

1.4.1 Coroutines or Tasks

1.4.2 Multi-threading

1.4.3 Green-threading

1.4.4 Hadoop and Julia

[1.5 AWS in Julia](#)

[1.5.1 AWSCore.jl](#)

[1.5.1.1 Credentials](#)

[1.5.1.2 S3 Containers](#)

[1.5.1.2.1 Listing buckets and Objects](#)

[1.5.1.2.2 Adding objects to existing bucket](#)

[1.5.2 AWS.jl](#)

[1.5.3 REST in julia](#)

[1.5.3.1 Genie.jl](#)

[1.5.3.1.1 Import Genie](#)

[1.5.3.1.2 API Backend](#)

[1.5.3.1.3 Adding Parameters to the URL](#)

[1.5.3.1.4 Create an app](#)

[1.5.3.1.5 Train Test Split Now we can extend functionality further by](#)

[1.5.3.1.6 Getting, Shuffling, Splitting data:](#)

[1.5.3.1.7 Extending functionality](#)

[1.6 Network](#)

[1.7 Optional Typing](#)

[2 SCALA](#)

[2.1 Scala for Cloud Computing](#)

[2.1.1 Language](#)

[2.1.1.1 Install Scala](#)

[2.1.1.2 Hello World! in Scala](#)

[2.1.1.3 Basics](#)

[2.1.1.4 Classes](#)

[2.1.1.4.1 Pattern matching](#)

[2.1.1.5 Scala as a functional language](#)

[2.1.1.6 Scala as a tool to create Domain Specific Languages\(DSL\)](#)

[2.1.1.6.1 Cloud computing with Scala and GridGain](#)

[2.1.2 Parallel programming in Scala](#)

[2.1.2.1 Parallel collections](#)

[2.1.2.2 Actor model](#)

[3 DEVTOOLS](#)

[3.1 Refcards](#)

[3.2 Virtual Box](#)

[3.2.1 Installation](#)

[3.2.2 Guest additions](#)

[3.2.3 Exercises](#)

[3.3 Vagrant](#) 🍀

[3.3.1 Installation](#)

[3.3.1.1 macOS](#)

[3.3.1.2 Windows](#) 0

[3.3.1.3 Linux](#) 0

[3.3.2 Usage](#)

[3.4 Packer](#) 0 🍀

[3.4.1 Installation](#)

[3.4.2 Usage](#)

[3.5 Ubuntu on an USB stick](#) 🍀

[3.5.1 Ubuntu on an USB stick for macOS via Command Line](#)

[3.5.1.1 Boot from the USB Stick](#)

[3.5.2 Ubuntu on an USB stick for macOS via GUI](#)

[3.5.2.1 Install Etcher](#)

[3.5.2.2 Prepare the USB stick](#)

[3.5.2.3 Etcher configuration](#)

[3.5.2.4 Write to the USB stick](#)

[3.5.3 Ubuntu on an USB stick for Windows 10](#) 0

[3.5.4 Exercise](#)

[3.6 GITHUB](#)

[3.6.1 Github](#) 🍀

[3.6.1.1 Overview](#)

[3.6.1.2 Upload Key](#)

[3.6.1.3 Fork](#)

[3.6.1.4 Rebase](#)

[3.6.1.5 Remote](#)

[3.6.1.6 Pull Request](#)

[3.6.1.7 Branch](#)

[3.6.1.8 Checkout](#)

[3.6.1.9 Merge](#)

[3.6.1.10 GUI](#)

[3.6.1.11 Windows](#)

[3.6.1.12 Git from the Commandline](#)

[3.6.1.13 Configuration](#)

[3.6.1.14 Upload your public key](#)

[3.6.1.15 Working with a directory that will be provided for you](#)

[3.6.1.16 README.yml and notebook.md](#)

[3.6.1.17 Contributing to the Document](#)

[3.6.1.17.1 Stay up to date with the original repo](#)

[3.6.1.17.2 Resources](#)

[3.6.1.18 Exercises](#)

[3.6.1.19 Github Issues](#)

[3.6.1.19.1 Git Issue Features](#)

[3.6.1.19.2 Github Markdown](#)

[3.6.1.19.2.1 Task lists](#)

[3.6.1.19.2.2 Team integration](#)

[3.6.1.19.2.3 Referencing Issues and Pull requests](#)

[3.6.1.19.2.4 Emojis](#)

[3.6.1.19.3 Notifications](#)

[3.6.1.19.4 cc](#)

[3.6.1.19.5 Interacting with issues](#)

[3.6.1.20 Glossary](#)

[3.6.1.21 Example commands](#)

[3.6.1.21.1 Local commands to version control your files](#)

[3.6.1.21.2 Interacting with the remote](#)

[3.6.2 Git Pull Request 🍀](#)

[3.6.2.1 Introduction](#)

[3.6.2.2 How to create a pull request](#)

[3.6.2.3 Fork the original repository](#)

[3.6.2.4 Clone your copy](#)

[3.6.2.5 Adding an upstream](#)

[3.6.2.6 Making changes](#)

[3.6.2.7 Creating a pull request](#)

[3.6.3 Tig 🍀](#)

[3.7 Linux Shell 🍀](#)

[3.7.1 History](#)

[3.7.2 Shell](#)

[3.7.3 The command man](#)

[3.7.4 Multi-command execution](#)

[3.7.5 Keyboard Shortcuts](#)

[3.7.6 bashrc, bash_profile or zprofile](#)

[3.7.7 Makefile](#)

[3.7.8 chmod](#)

[3.7.9 Exercises](#)

[3.8 Secure Shell](#) 🍀

[3.8.1 ssh-keygen](#)

[3.8.2 ssh-add](#)

[3.8.3 SSH Add and Agent](#)

[3.8.3.1 Using SSH on Mac OS X](#)

[3.8.3.2 Using SSH on Linux](#)

[3.8.3.3 Using SSH on Raspberry Pi 3/4](#)

[3.8.3.4 Accessing a Remote Machine](#)

[3.8.4 SSH Port Forwarding](#) 📺

[3.8.4.1 Prerequisites](#)

[3.8.4.2 How to Restart the Server](#)

[3.8.4.3 Types of Port Forwarding](#)

[3.8.4.4 Local Port Forwarding](#)

[3.8.4.5 Remote Port Forwarding](#)

[3.8.4.6 Dynamic Port Forwarding](#)

[3.8.4.7 ssh config](#)

[3.8.4.8 Tips](#)

[3.8.4.9 References](#)

[3.8.5 SSH to FutureSystems Resources](#) 🍀

[3.8.5.1 Testing your FutureSystems ssh key](#)

[3.8.6 Exercises](#) 🍀

[4 DEVOPS](#)

[4.1 DevOps](#) 🍀

[4.1.1 References](#)

[4.2 Travis](#) 🍀

[4.2.1 Exercises](#)

[4.2.2 Resources](#)

[4.3 Ansible](#) 🍀

[4.3.1 Introduction to Ansible](#)

[4.3.1.1 Prerequisite](#)

[4.3.1.2 Setting up a playbook](#)

[4.3.1.3 Run the playbook](#)

[4.3.2 Ansible Roles](#)

[4.3.3 Using Variables](#)

[4.3.4 Ansible Galaxy](#)

- [4.3.4.1 Ansible Galaxy helloworld](#)
- [4.3.5 A Complete Ansible Galaxy Project](#)
 - [4.3.5.1 Ansible: Write a Playbooks for MongoDB](#)
 - [4.3.5.2 First playbook for MongoDB Installation](#)
 - [4.3.5.2.1 Enabling Root SSH Access](#)
 - [4.3.5.2.2 Hosts and Users](#)
 - [4.3.5.2.3 Tasks](#)
 - [4.3.5.2.4 Module apt_key: add repository keys](#)
 - [4.3.5.2.5 Module apt_repository: add repositories](#)
 - [4.3.5.2.6 Module apt: install packages](#)
 - [4.3.5.2.7 Module service: manage services](#)
 - [4.3.5.2.8 The Full Playbook](#)
 - [4.3.5.2.9 Running a Playbook](#)
 - [4.3.5.2.10 Sanity Check: Test MongoDB](#)
 - [4.3.5.2.11 Terms](#)
 - [4.3.5.2.12 Reference](#)
- [4.3.6 Exercise](#)

[5 DRAFTS](#)

[5.1 DATA SERVICES](#)

[5.1.1 Box](#)

[5.1.1.1 Limitations](#)

[5.1.1.2 Creating an app](#)

[5.1.1.2.1 Authentication with JWT](#)

[5.1.1.3 Box Methods](#)

[5.1.1.3.1 Get information about a Box object](#)

[5.1.1.3.2 Folders](#)

[5.1.1.3.3 Uploading files](#)

[5.1.1.3.4 File upload errors](#)

[5.1.1.3.5 Deleting, copying, and downloading files](#)

[5.1.1.3.6 Searching](#)

[5.1.1.3.7 Shared links](#)

[5.1.1.4 Project management](#)

[5.1.1.4.1 Collaborations](#)

[5.1.1.4.2 Groups](#)

[5.1.1.4.3 Tasks](#)

[5.1.1.5 Pybox](#)

[5.1.1.6 Cloudmesh](#)

[5.1.1.6.1 Get](#)

[5.1.1.6.2 Put](#)

[5.1.1.6.3 Search](#)

[5.1.1.6.4 List](#)

[5.1.1.6.5 Create a directory](#)

[5.1.1.6.6 Delete](#)

[5.1.2 AWS DocumentDB \(hid: sp19-516-134\) 🍀](#)

[5.1.2.1 Amazon DocumentDB Features and Benefits](#)

[5.1.2.2 AWS Document database Pricing](#)

[5.1.2.3 How to provision Document database](#)

[5.1.2.3.1 Step 1: Login to the AWS console.](#)

[5.1.2.3.2 Step 2: Click on Create Amazon DocumentDB cluster button.](#)

[5.1.2.3.3 Step 3: Create the required configuration.](#)

[5.1.2.3.4 Step 3: Cluster creation process](#)

[5.1.2.3.5 Step 4: Cluster summary and connection information from Mongo and Application.](#)

[5.1.2.3.6 Step 5: Cluster Details.](#)

[5.1.2.3.7 Step 6: Cloudwatch Information on the clusters.](#)

[5.1.2.4 How to connect to Document Database](#)

[5.1.3 Amazon Aurora DB 🍀](#)

[5.1.3.1 Amazon Aurora Features and Benefits](#)

[5.1.3.2 Amazon Aurora Pricing](#)

[5.1.3.3 How to provision Aurora database](#)

[5.1.3.3.1 Step 1: Login to the AWS console.](#)

[5.1.3.3.2 Step 2: Click on Create Database](#)

[5.1.3.3.3 Step 3: Select Amazon Aurora Engine](#)

[5.1.3.3.4 Step 4: Aurora Configuration and Settings](#)

[5.1.3.3.4.1 Configuration](#)

[5.1.3.3.4.2 Settings](#)

[5.1.3.3.4.3 Step 5: Configure Advance Settings - Network and Security](#)

[5.1.3.3.4.4 Step 6: Configure Advance Settings - Database Options](#)

[5.1.3.3.4.5 Step 7: Configure Advance Settings - Encryption and Failover](#)

[5.1.3.3.4.6 Step 8: Configure Advance Settings - Backtrack and](#)

[Monitoring](#)

[5.1.3.3.4.7 Step 9: Configure Advance Settings - Log Exports and Maintenance](#)

[5.1.3.3.4.8 Step 10: Submit Advance Settings Page](#)

[5.1.3.3.5 Step 11: Wait for DB creation to complete](#)

[5.1.3.4 How to retrieve Aurora database connection details](#)

[5.1.3.5 Other details available in Aurora DB Console](#)

[5.1.3.5.1 Monitoring](#)

[5.1.3.5.2 Logs and Events](#)

[5.1.3.5.3 Configuration](#)

[5.1.3.5.4 Tag](#)

[5.1.3.6 Update Aurora DB](#)

[5.1.3.6.1 Modify](#)

[5.1.3.6.2 Reboot](#)

[5.1.3.6.3 Delete](#)

[5.1.3.6.4 Failover](#)

[5.1.3.6.5 Take Snapshot](#)

[5.1.3.7 Example](#)

[5.1.3.8 Exercises](#)

[5.1.3.9 References](#)

[5.2 AI](#)

[5.2.1 Artificial Intelligence Service with REST](#) 🍀

[5.2.1.1 AI and ML offerings by Cloud service providers](#)

[5.2.1.2 Image Analysis using Google Cloud Vision API](#)

[5.2.1.3 Naive Bayes Algorithm for Text classification](#)

[5.3 LINUX](#)

[5.3.1 Windows Subsystem for Linux](#) 🍀

[5.4 DEVOPS](#)

[5.4.1 Infrastructure as Code \(IaC\)](#) 🍀

[5.4.1.1 Learning Objectives](#)

[5.4.1.2 Introduction to IaC](#)

[5.4.1.3 How IaC is related to DevOps](#)

[5.4.1.4 How IaC tools differs from Configuration Management Tools, and how it is related](#)

[5.4.1.5 Listing of IaC Tools](#)

[5.4.1.6 Advantages of IaC](#)

[5.4.1.7 Further Reading](#)

[5.5 OTHER](#)

[5.5.1 Amazon Elastic Beanstalk](#)

[5.5.1.1 Amazon Elastic Beanstalk Pricing](#)

[5.5.1.2 How to provision a Amazon Elastic Beanstalk Environment for Quick Start](#)

[5.5.1.2.1 Step 1: Login to the AWS console.](#)

[5.5.1.2.2 Step 2: Click on Create New Application](#)

[5.5.1.2.3 Step 3: Select Base Configuration](#)

[5.5.1.2.4 Step 4: Click on Configure More Options](#)

[5.5.1.2.5 Step 5: Select High Availability as Config Preset](#)

[5.5.1.2.6 Step 6: Map VPC](#)

[5.5.1.2.6.1 Load Balancer settings](#)

[5.5.1.2.6.2 Instance settings](#)

[5.5.1.2.7 Step 7: Launch Application](#)

[5.5.1.2.8 Step 8: Verify application URL](#)

[5.5.1.3 Explore Amazon Elastic Beanstalk Features](#)

[5.5.1.3.1 Beanstalk Deployment](#)

[5.5.1.3.1.1 Tracking Deployments](#)

[5.5.1.3.2 Beanstalk Env management](#)

[5.5.1.3.3 Beanstalk Logs](#)

[5.5.1.3.4 Beanstalk Instance Health](#)

[5.5.1.3.5 Beanstalk Environment Monitoring and Alarms](#)

[5.5.1.3.6 Beanstalk Patching Logs](#)

[5.5.1.3.7 Beanstalk Event Logs](#)

[5.5.1.3.8 Beanstalk Tags](#)

[5.5.1.3.9 Beanstalk Configuration Management](#)

[5.5.1.3.9.1 Software](#)

[5.5.1.3.9.2 Instances](#)

[5.5.1.3.9.3 Capacity](#)

[5.5.1.3.9.4 Load Balancer](#)

[5.5.1.3.9.5 Rolling updates and deployments](#)

[5.5.1.3.9.6 Security](#)

[5.5.1.3.9.7 Monitoring](#)

[5.5.1.3.9.8 Managed Updates](#)

[5.5.1.3.9.9 Notifications](#)

[5.5.1.3.9.10 Network](#)

[5.5.1.3.9.11 Database](#)

[5.5.1.4 Access Beanstalk using python](#)

[5.5.1.5 Exercises](#)

[5.5.1.6 References](#)

[5.5.2 Compliance and Grid Computing](#)  

[5.5.3 Visualization](#)  

[5.5.3.1 Chart Types](#)

[5.5.3.2 D3 Gallery](#)

[5.5.3.3 Matplot Gallery](#)

[5.5.3.4 Bokeh Gallery](#)

[5.5.3.5 R Gallery](#)

[5.5.3.6 Gnuplot](#)

[5.5.3.7 React](#)

[5.5.3.8 Tableau](#)

[5.5.4 Distributed Message Queues](#) 

[5.5.4.1 AMPQ](#)

[5.5.4.2 screenshot rename automator](#) 

[6 ISSUES](#)

[6.1 Github Issues](#) 

[6.1.1 Internal Issues](#)

[6.1.2 Open Sections](#)

[6.1.3 Open Chapters](#)

[6.1.4 Assigned Sections](#)

[6.1.5 Assigned Chapters](#)

[6.1.6 Open Projects](#)

[6.1.7 Assigned Projects](#)

[7 WEEKLY](#)

[7.1 Weekly Activities](#) 

[7.1.1 Week 1: Activities Aug 26 - 30](#)

[7.1.2 Week 2: Aug 30 - Sep 6](#)

[7.1.2.1 Development machine](#)

[7.1.2.2 Data Center](#)

[7.1.2.2.1 Datacenter Table](#)

[7.1.2.2.2 Data Center sections](#)

[7.1.2.2.3 Datacenter Exercises](#)

[7.1.2.3 Remarks: Sections](#)

[7.1.2.4 Python till rest of the semester](#)

[7.1.3 Week 3: Sep 6 - Sep 13](#)

- [7.1.3.1 Cloud Architectures](#)
- [7.1.3.2 REST services](#)
- [7.1.4 Week 4: Sep 13 - Sep 20](#)
 - [7.1.4.1 Github as REST service](#)
 - [7.1.4.2 Practical OpenAPI](#)
- [7.1.5 Week 5: Sep 20 - Sep 27](#)
- [7.1.6 Week 5 and 6: Sep 27 - Oct 4 - Oct 11](#)
 - [7.1.6.1 Sections and Chapters](#)
 - [7.1.6.2 IaaS](#)
 - [7.1.6.3 Libcloud and MongoDB](#)
- [7.1.7 Week 7: Oct 11 - Oct 18](#)
 - [7.1.7.1 MapReduce](#)
- [7.1.8 Fall Break: Oct 18 - Oct 20](#)
- [7.1.9 Week 8: Oct 20 - Oct 25](#)
 - [7.1.9.1 Hadoop and Spark This week you will further look into MapReduce](#)
- [7.1.10 Week 9: Oct 25 - Nov 1](#)
 - [7.1.10.1 Containers](#)
- [7.1.11 Week 10: Docker Cluster Nov 1 - Nov 8](#)
- [7.1.12 Week 11: Kubernetes Nov 8 - Nov 15](#)
- [7.1.13 Week 12: Kubernetes Nov 15 - Nov 22](#)
- [7.1.14 Thanksgiving break Nov 24 - Dec 1](#)
- [7.1.15 Project Due Date Dec 1](#)
- [7.1.16 Week 13: Go or Julia Dec 1 - Dec 6](#)
- [7.1.17 Week 14: Dec 6 - Dec 13](#)
- [7.1.18 Week 15: Dec 13 - 20](#)

[8 QUICKSTART](#)

[8.1 Overview](#)

- [8.1.1 Requirements](#)
- [8.1.2 Time Commitment](#)
- [8.1.3 Course Material List](#)
- [8.1.4 Help](#)
- [8.1.5 How to Take this Class](#)
- [8.1.6 Assignments](#)
 - [8.1.6.1 Section](#)
 - [8.1.6.2 Chapter with Example](#)
 - [8.1.6.3 Project](#)

- [8.1.6.3.1 License](#)
- [8.1.6.3.2 Project Report](#)
- [8.1.6.3.3 Project Code](#)
- [8.1.6.3.4 Project Data](#)
- [8.1.6.3.5 Work Breakdown](#)
- [8.1.6.3.6 Bibliography](#)

[8.1.6.4 Project Deliverables](#)

[8.1.7 Submission of sections and chapters and projects](#)

[8.1.8 Participation](#)

[8.2 Assignments](#)

[8.2.1 Account Creation](#)

[8.2.2 Sections, Chapters with Examples](#)

[8.2.3 Mini Projects that could Substitute a Chapter](#)

[8.2.4 Project](#)

[8.2.4.1 Project Deliverables](#)

[8.2.5 Project: Virtual Cluster](#)

[8.2.6 Submission of sections and chapters and projects](#)

[8.2.7 Participation](#)

[8.3 Course Syllabus Tables](#)

[8.3.1 Proposed Lecture Timeline](#)

[8.3.2 Assignments Timeline](#)

[8.3.3 Group Breakdown Checkpoint](#)

[9 OVERVIEW](#)

[9.1 Organization](#)

[9.1.1 First Week](#)

[9.1.2 IoT Hardware](#)

[9.1.3 Access to Clouds](#)

[9.1.4 Using Your Own Computer](#)

[9.1.4.1 Self Discipline](#)

[9.1.4.2 Fun](#)

[9.1.4.3 Uniqueness](#)

[9.1.4.4 Continuation](#)

[9.1.5 Parallel Tracks](#)

[9.1.5.1 Track 1: Practice](#)

[9.1.5.2 Track 2: Theory](#)

[9.1.5.3 Track 3: Writing](#)

[9.1.5.4 Track 4: Project](#)

[9.1.6 Plagiarism](#) 

[9.2 Research Interests](#) 

[9.3 e516: Engineering Cloud Computing](#) 

[9.3.1 Course Description](#)

[9.3.2 Course Objectives](#)

[9.3.3 Learning Outcomes](#)

[9.4 Syllabus](#)

[9.4.1 Assessment](#)

[9.4.1.1 Incomplete](#)

[9.5 Course Policies](#) 

[9.5.1 Discussion via Piazza](#)

[9.5.2 Managing Your Own Calendar](#)

[9.5.3 Online and Office Hours](#)

[9.5.3.1 Office Hour Calendar](#)

[9.5.4 Class Material](#)

[9.5.5 HID](#)

[9.5.6 Class Directory](#)

[9.5.7 Notebook](#)

[9.5.8 Blog](#)

[9.5.9 Waitlist](#)

[9.5.10 Registration](#)

[9.5.11 Auditing the class](#)

[9.5.12 Resource restrictions](#)

[9.5.13 Incomplete](#)

[9.5.13.1 Exercises](#)

[9.6 E516 Summary](#) 

[9.6.1 Introduction](#)

[9.6.1.1 Research participation](#)

[9.6.2 Class communication](#)

[9.6.2.1 Topics covered in this class](#)

[9.6.3 Assignments](#)

[9.6.4 Scientific Writing](#)

[9.7 Example Artifacts](#) 

[9.7.1 Technology Summaries](#)

[9.7.2 Chapters](#)

[9.7.3 Project Reports](#)

[10 GRAPHICS](#)

[10.1 Graphviz](#)

[10.1.1 Installation](#)

[10.1.2 Usage](#)

[10.1.2.1 The Dot Format](#)

[10.1.3 Exercise](#)

[11 ART](#)

[11.1 Sentient Architecture](#)

[11.1.1 Introduction](#)

[11.1.2 Existing Deployments](#)

[11.1.3 Impact](#)

[11.1.4 Integration into Cloud Computing and Big Data](#)

[11.1.5 Development](#)

[11.1.5.1 Snowwhite and the Seven⁺¹ Dwarfs](#)

[11.1.5.2 Luddy Hall Installation](#)

[11.1.6 Sentient Cloudmesh](#)

[11.1.6.0.0.1 Snowwhite and the Seven⁺¹ Dwarfs](#)

[11.1.6.0.0.2 Luddy Hall](#)

[11.1.7 Alternative Boards](#)

[11.1.7.1 Mega 2560](#)

[11.1.7.2 Programming with Teensy](#)

[11.1.8 Exercises](#)

[12 OTHER](#)

[12.0.1 Fall 2018](#)

[12.0.1.1 Registrar](#)

[12.0.1.2 ENGR-E 516 Residential](#)

[12.0.1.3 ENGR-E 516 Online](#)

[12.0.1.4 ENGR-E 416](#)

[12.0.1.5 ENGR-E 416](#)

[12.0.1.6 CSCI-B 649 Topics in Systems](#)

[12.0.1.7 ENGR-E 534 Residential](#)

[12.0.1.8 INFO-I 523 Residential](#)

[12.0.1.9 INFO-I 523 Online](#)

[12.0.1.10 INFO-I 423 Residential](#)

[12.0.2 Syllabus](#)

[12.0.3 Assignments](#)

[12.0.3.1 Terminology](#)

[12.0.3.2 Due dates](#)

[12.0.3.3 Assignment 0 - Organization and Communication](#)

[12.0.3.3.1 Account setup and survey](#)

[12.0.3.3.2 Class Computer Setup](#)

[12.0.3.3.3 Outcomes](#)

[12.0.3.4 Assignment 1 - Technology](#)

[12.0.3.4.1 Outcomes](#)

[12.0.3.5 Assignment 2 - Programming Assignment & Chapter](#)

[12.0.3.5.1 Outcomes](#)

[12.0.3.6 Assignment 3 - Project](#)

[12.0.3.6.1 Class: E516 or E616](#)

[12.0.3.6.1.1 Deliverables](#)

[12.0.3.6.2 Class: i423 i523 or E534 or i524](#)

[12.0.3.6.2.1 Deliverables](#)

[12.0.3.6.3 Class: E222](#)

[12.0.3.6.3.1 Deliverables](#)

[12.1 Grading](#) 

[12.1.1 Grades on Canvas](#)

[12.1.2 Discussion about Grades](#)

[12.1.3 Project Ideas](#) 

[12.1.3.1 Project Data Restrictions](#)

[12.1.3.2 Register Your Idea](#)

[12.1.3.3 Example Ideas](#)

[12.1.3.3.1 Project Type A: NIST Rest Services Project](#)

[12.1.3.3.2 Project Type B: Reproducible Raspberry PI Projects](#)

[12.1.3.3.3 Project Type C: Reproducible Data Analysis](#)

[12.1.3.3.4 Project Type D: Define Your Own](#)

[12.1.3.4 Special Projects](#)

[12.1.3.4.1 Project Cloud Security](#)

[12.1.3.4.2 Hadoop 3.0](#)

[12.1.4 Piazza](#) 

[12.1.4.1 Access to Piazza from Canvas](#)

[12.1.4.2 Piazza for 516](#)

[12.1.4.3 Piazza for other classes](#)

[12.1.4.3.1 Situation: You have never logged into piazza](#)

[12.1.4.3.2 Situation: You have logged into piazza and used your default IU e-mail](#)

[12.1.4.3.3 Situation: You have logged into piazza and you used another non IU e-mail](#)

[12.1.4.3.4 Situation: You have multiple accounts in piazza](#)

[12.1.4.4 Verify you are on Piazza via a post](#)

[12.1.4.5 Making Piazza Work](#)

[12.1.4.6 Towards good questions](#)

[12.1.4.7 Guide on how to ask good questions](#)

[12.1.4.8 Piazza class Links](#)

[12.1.4.8.1 Current Classes](#)

[12.1.4.8.2 Previous Classes](#)

[12.1.4.9 Piazza Curation](#)

[12.1.4.10 Read the Originals, not just the e-mail](#)

[12.1.4.11 Exercises](#)

[12.1.5 Class Git](#) 🍀

[13 APPENDIX](#)

[13.1 FAQ](#)

[13.1.1 FAQ: General](#) 🍀

[13.1.1.1 Can I assume that all information is in the FAQ to do the class?](#)

[13.1.1.2 Piazza](#)

[13.1.1.2.1 Why are some FAQs that are on piazza not here?](#)

[13.1.1.3 How do I find all FAQ's in Piazza?](#)

[13.1.1.4 Has SOIC computers I can use remotely?](#)

[13.1.1.5 When contributing to the book my name is not listed](#)

[13.1.1.6 How to read the technical sections of the lecture notes](#)

[13.1.1.7 How to check if a yaml file is valid?](#)

[13.1.1.8 Download the ePub frequently](#)

[13.1.1.9 Spelling of filenames in github](#)

[13.1.1.10 How to open the ePub from Github?](#)

[13.1.1.11 Assignment Summary](#)

[13.1.1.12 Auto 80 char](#)

[13.1.1.13 Useful FAQs for residential and online students](#)

[13.1.1.14 What if i committed a wrong file to github, a.g. a private key?](#)

[13.1.2 FAQ: 516](#) 🍀

[13.1.2.1 Why have the lectures been removed from the resources section?](#)

- [13.1.2.2 Opening ePub properly in Linux](#)
- [13.1.2.3 Microsoft Edge for ePub](#)
- [13.1.2.4 Project format: Markdown allowed](#)
- [13.1.2.5 pull request in documents from commandline](#)
- [13.1.2.6 pyenv installed but can not find python](#)
- [13.1.2.7 ssh add/keychain on OSX](#)
- [13.1.2.8 Check into github early](#)
- [13.1.2.9 I want to start my project now](#)
- [13.1.2.10 Mac with external hard drive](#)
- [13.1.2.11 Submitting a section](#)
- [13.1.2.12 Where to post comprehensions?](#)
- [13.1.2.13 Communication of sections into the README.yml](#)
- [13.1.2.14 Question on adding new sections to the book and github protocol](#)
- [13.1.2.15 Where to post references](#)
- [13.1.2.16 where are the lectures?](#)
- [13.1.2.17 Swagger UI](#)
- [13.1.2.18 Image file names \(IMPORTANT\)](#)
- [13.1.2.19 Report template](#)
- [13.1.2.20 improved ePub viewing of ePub documents](#)
- [13.1.2.21 empty lines in markdown](#)
- [13.1.2.22 Linux ePub Reader \(followup\)](#)
- [13.1.2.23 auto 80 char](#)
- [13.1.2.24 what if i committed a wrong file to github, a.g. a private key?](#)
- [13.1.2.25 markdown and bibtex](#)

[13.2 Amazon Web Service Products 🍀](#)

- [13.2.1 Compute](#)
- [13.2.2 Storage](#)
- [13.2.3 Databases](#)
- [13.2.4 Migration](#)
- [13.2.5 Networking & Content Delivery](#)
- [13.2.6 Developer Tools](#)
- [13.2.7 Management Tools](#)
- [13.2.8 Media Services](#)
- [13.2.9 Security, Identity & Compliance](#)
- [13.2.10 Machine Learning](#)

- [13.2.11 Analytics](#)
- [13.2.12 Mobile](#)
- [13.2.13 AR & VR](#)
- [13.2.14 Application Integration](#)
- [13.2.15 Customer Engagement](#)
- [13.2.16 Business Productivity](#)
- [13.2.17 Desktop & App Streaming](#)
- [13.2.18 Internet of Things](#)
- [13.2.19 Game Development](#)
- [13.2.20 AWS Marketplace Software](#)
- [13.2.21 AWS Cost Management](#)
- [13.2.22 Exercise](#)

[13.3 Microsoft Azure and Cloud Products](#)

- [13.3.1 AI + Machine Learning](#)
- [13.3.2 Analytics](#)
- [13.3.3 Compute](#)
- [13.3.4 Containers](#)
- [13.3.5 Databases](#)
- [13.3.6 Developer Tools](#)
- [13.3.7 DevOps](#)
- [13.3.8 Identity](#)
- [13.3.9 Integration](#)
- [13.3.10 Internet of Things](#)
- [13.3.11 Management Tools](#)
- [13.3.12 Media](#)
- [13.3.13 Microsoft Azure Stack](#)
- [13.3.14 Migration](#)
- [13.3.15 Mobile](#)
- [13.3.16 Networking](#)
- [13.3.17 Security](#)
- [13.3.18 Storage](#)
- [13.3.19 Web](#)

[13.4 FutureSystems](#)

- [13.4.1 FutureSystems evolved from FutureGrid](#)
- [13.4.2 Creating Portal Account](#)
- [13.4.3 SSH Key Generation using ssh-keygen command](#)
- [13.4.4 Shell Access via SSH](#)

[13.4.5 Advanced SSH](#)

[13.4.6 SSH Key Generation via putty](#)

[13.4.7 FutureSystems Facilities](#)

[13.4.7.1 Bravo](#)

[13.4.7.2 Delta](#)

[13.4.7.3 Echo](#)

[13.4.7.4 Juliet](#)

[13.4.7.5 Romeo](#)

[13.4.7.6 Tango](#)

[13.4.7.7 Tempest](#)

[13.4.7.8 Victor](#)

[13.4.7.9 PI Cluster](#)

[13.5 CHAMELEON](#)

[13.5.1 Resources](#)

[13.5.1.1 Outages](#)

[13.5.1.2 Account Creation](#)

[13.5.1.3 Join a Project](#)

[13.5.1.4 Usage Restriction](#)

[13.5.2 Chameleon Cloud Hardware](#)

[13.5.2.1 Standard Cloud Units](#)

[13.5.2.2 Network](#)

[13.5.2.3 Shared Storage](#)

[13.5.2.4 Heterogeneous Compute Hardware](#)

[13.5.3 Chameleon Cloud Charge Rates](#)

[13.5.3.1 Service Units](#)

[13.5.3.2 Project Allocation Size](#)

[13.5.4 Getting Started on Chameleon Cloud](#)

[13.5.4.1 Step 1: Create a Chameleon account](#)

[13.5.4.2 Step 2: Create or join a project](#)

[13.5.4.3 Step 3: Start using Chameleon](#)

[13.5.5 OpenStack Virtual Machines](#)

[13.5.5.1 Web Interface](#)

[13.5.5.1.1 Managing Virtual Machine Instances](#)

[13.5.5.1.2 Snapshots](#)

[13.5.5.1.3 Firewall \(Access Security\)](#)

[13.5.5.2 OpenStack REST Interfaces](#)

[13.5.5.3 Downloading and uploading data](#)

[13.5.6 Cloudmesh OpenStack Command Line Interface](#)

[13.5.6.1 Instalation of Cloudmesh Client](#)

[13.5.6.2 Floating IP Address](#)

[13.5.7 OpenStack Horizon Graphical User Interface](#)

[13.5.7.1 Configure resources](#)

[13.5.7.2 Interact with resources](#)

[13.5.7.2.1 Snapshot an instance](#)

[13.5.7.3 Use FPGAs](#)

[13.5.7.4 Next Step](#)

[13.5.8 OpenStack HEAT](#)

[13.5.8.1 Supporting Complex Appliances](#)

[13.5.8.2 Chameleon Appliance Catalog](#)

[13.5.8.3 Deployment](#)

[13.5.8.4 Heat Template](#)

[13.5.8.5 Customizing an existing template](#)

[13.5.8.6 Writing a new template](#)

[13.5.8.6.1 Heat template version](#)

[13.5.8.6.2 Description](#)

[13.5.8.6.3 Resources](#)

[13.5.8.6.4 Parameters](#)

[13.5.8.6.5 Outputs](#)

[13.5.8.7 Sharing new complex appliances](#)

[13.5.8.8 Advanced topics](#)

[13.5.8.8.1 All-to-all information exchange](#)

[13.5.9 Openstack Bare Metal](#)

[13.5.10 Chameleon Cloud Frequently Asked Questions](#)

[13.5.10.1 Appliances](#)

[13.5.10.1.1 What is an appliance?](#)

[13.5.10.1.2 What is the Appliance Catalog?](#)

[13.5.10.1.3 How do I publish an appliance in the Appliance Catalog?](#)

[13.5.10.1.4 How can I manage an appliance on Appliance Catalog?](#)

[13.5.10.1.5 Why are there different image IDs for the same appliance?](#)

[13.5.10.1.6 Can I use another operating system on bare-metal?](#)

[13.5.10.2 Bare Metal Troubleshooting](#)

[13.5.10.2.1 Why are my Bare Metal instances failing to launch?](#)

[13.5.10.3 OpenStack KVM Troubleshooting](#)

[13.5.10.3.1 Why are my OpenStack KVM instances failing to launch?](#)

[13.5.10.3.2 Why can I not ping or SSH to my instance?](#)

[13.6 GLOSSARY](#)

[13.6.1 Glossary](#)

[13.6.1.1 VM and Container](#)

[13.6.1.2 Network](#)

[13.6.1.3 Storage](#)

[13.6.2 Creating the ePubs from source](#)

[13.6.2.1 Docker](#)

[13.6.2.1.1 Using OSX](#)

[13.6.2.1.2 Using the Docker Image](#)

[13.6.2.2 Using the Native System](#)

[13.6.2.3 Using Vagrant](#)

[13.6.2.4 Creating a Book](#)

[13.6.2.5 Publishing the Book to GitHub](#)

[13.6.2.6 Creating Drafts](#)

[13.6.2.7 Creating a New Book](#)

[13.6.2.8 Managing Images](#)

[13.6.2.9 Managing Refernces](#)

[14 REFERENCES](#)

1 JULIA

1.1 INTRODUCTION TO JULIA FOR HIGH PERFORMANCE COMPUTING



Learning Objectives

- Get up and running with Julia
 - Learn basic AWS API functionality in Julia
 - Learn about REST services in Julia
 - Interact with external programs in Julia
-

The Julia developers claim: “Julia walks like Python and runs like C.” [1]

Because Julia is a relatively new programming language and documentation is sparse, much of this chapter has been adapted from <https://julialang.org> [1]

1.1.1 Introduction

Julia is a high-performance computing language designed by Alan Edelman, Jeff Bezanson, Stefan Karpinski, and Viral Shah. Released in 2012, Julia was aimed at the frustration of having to use multiple languages for various computing tasks; Matlab for linear algebra, R for statistics, Python for web development, etc. Karpinski and Shah teamed up with Bezanson to develop a high-level, human readable, all-purpose programming language that didn’t require translation into lower languages like C or Java. “Using LLVM, a compiler developed by University of Illinois at Urbana-Champaign and enhanced by the likes of Apple and Google, Karpinski and company built the language so that it compiles straight to machine code on the fly, as it runs” [2]. Users will notice that the first time a piece of code runs in Julia, it takes some time to compile. Subsequent runs are extremely efficient.

“Julia features optional typing, multiple dispatch, and good performance,

achieved using type inference and just-in-time (JIT) compilation, implemented using LLVM” [1].

Julia is a dynamically-typed language, much like Python or R, which tend to be more human-readable than lower-level languages such as Java or C. Unfortunately, due to the tasks of translating high-level into low-level, the performance cost can be high. Depending on the task, dynamically-typed languages can be slow, especially for memory-intensive operations. Conversely, languages such as C are compiled or statically-typed languages are fast, but much less human-readable. Julia provides the best of both worlds. “Julia is fast because of careful language design and the right combination of carefully chosen technologies that work very well with each other” [3].

Julia is a high-performance computing language designed by Alan Edelman, Jeff Bezanson, Stefan Karpinski, and Viral Shah. Released in 2012, Julia was aimed at the frustration of having to use multiple languages for various computing tasks; Matlab for linear algebra, R for statistics, Python for web development, etc. Karpinski and Shah teamed up with Bezanson to develop a high-level, human readable, all-purpose programming language that didn’t require translation into lower languages like C or Java. “Using LLVM, a compiler developed by University of Illinois at Urbana-Champaign and enhanced by the likes of Apple and Google, Karpinski and company built the language so that it compiles straight to machine code on the fly, as it runs” [2].

“Julia features optional typing, multiple dispatch, and good performance, achieved using type inference and just-in-time (JIT) compilation, implemented using LLVM” [1].

Julia is a dynamically-typed language, much like Python or R, which tend to be more human-readable than lower-level languages such as Java or C. Unfortunately, due to the tasks of translating high-level into low-level, the performance cost can be high. Depending on the task, dynamically-typed languages can be slow, especially for memory-intensive operations. Conversely, languages such as C are compiled or statically-typed languages are fast, but much less human-readable. Julia provides the best of both worlds. “Julia is fast because of careful language design and the right combination of carefully chosen technologies that work very well with each other” [3].

1.1.1.1 Development Environments

In line with Julia's hybrid DNA of compiled and dynamic languages, a user can either type commands into a shell, or create programs in a file with a `.jl` extension, and call the files externally. Other editors and IDEs are also available.

1.1.2 Module Management

Julia provides module and package management similar to Python's `pip` with the `Pkg.add("")` function. [4]. To use a package in a working environment, simply issue the command `using <Package>`. For instance, to add a graphics package called Winston and plot 100 random numbers, execute the following commands:

Julia provides module and package management similar to Python's `pip` with the `Pkg.add("")` function. [4]. To use a package in a working environment, simply issue the command `using <Package>`. For instance, to add a graphics package called Winston and plot 100 random numbers, execute the following commands:

```
julia> Pkg.add("Winston")
julia> using Winston
julia> plot(rand(100))
```

1.1.3 Multiple Dispatch

This section is still under development.

1.1.4 Parallel Language Constructs

This section is still under development.

1.1.5 Interfacing with the System

This section is still under development.

Later sections cover module and package management and development in greater detail.

1.1.6 Installing Julia

🕒 this section is outdated

To find the correct installation file for a given platform, visit <https://julialang.org/downloads/>. There are several ways to run and interact with Julia:

- In the terminal using the Julia REPL
- In a browser using JuliaBox with Jupyter Notebooks
- Using Docker images provided on DockerHub and provided by the Docker community
- By subscribing to JuliaPro by Julia Computing, which includes Juno IDE, discussed later in this chapter.

1.1.6.1 Julia REPL

1.1.6.1.1 Linux

First, check whether the required dependencies are installed: <https://github.com/JuliaLang/julia#required-build-tools-and-external-libraries>

On Ubuntu, you can easily install the required packages using the following command [1]:

```
$ sudo apt-get install build-essential libatomic1 python gfortran perl wget m4  
$ cmake pkg-config
```

To install Julia locally, first download and extract the appropriate 32 or 64-bit binary installation package at <https://julialang.org/downloads/> into a directory of your choice. Extract the tarball or gzip binary file.

```
$ wget https://julialang-s3.julialang.org/bin/linux/x64/1.0/julia-1.0.1-linux-x86_64.tar.gz  
$ tar xvfz julia-1.0.1-linux-x86_64.tar.gz
```

You can then add Julia home to your PATH, or include a soft reference as follows:

```
$ export PATH="$(pwd)/julia:$PATH"
```

Alternatively, bring the REPL into scope by editing your `/etc/environment` file. Finally, you can create a “hard” link with the following command:

```
$ sudo ln -s ~/JuliaPro-0.6.2.1/Julia/bin/julia /usr/local/bin/julia
```

Where the last `julia` is input into bash to start the REPL. Creating a link will enable you to call different versions of Julia directly from the bash or shell. For instance, if both Julia 0.6 and Julia 1.0 are required, it is possible to create separate links to both as follows:

```
# hard link to julia 0.6
$ sudo ln -s ~/JuliaPro-0.6.2.1/Julia/bin/julia /usr/local/bin/julia0.6

# hard link to julia 1.0
$ sudo ln -s ~/JuliaPro-1.1.3/Julia/bin/julia /usr/local/bin/julia1.0
```

[1]

1.1.6.1.2 Windows

Download the appropriate installer file from <https://julialang.org/downloads/> and run the installer. You should now be able to activate Julia in your Bash or Shell session by typing:

```
$ julia
```

1.1.6.2 JuliaBox

Finally, the simplest method for developing with Julia is by interacting directly in a browser using <https://www.Juliabox.com>. This method requires no installation, and provides the familiar look and feel of Jupyter Notebooks without installing any additional software. Further, this option allows creation and local storage of Jupyter Notebooks for later use in JuliaBox. Type the URL into a browser, log in or create an account, and select the tier or subscription level to begin.

1.1.7 Julia IDE

As stated in earlier sections, there are several development environments and editors to use for Julia. For purposes of the following sections, Juno (Atom) will be the primary IDE.

1.1.7.1 Install Atom

Assuming Julia is already installed, download and install the appropriate Atom

distribution from <https://atom.io/>. Ensure that the version is up-to-date (v1.23 or higher).

1.1.7.2 Install Juno

To install Juno within Atom, go to the Settings > Install panel. Type `uber-juno` into the search bar and hit enter. Find the `uber-juno` package and install. After Atom installs Juno, simply restart Atom. For further instructions, visit the installation page at <http://docs.junolab.org/latest/man/installation/>. Use the REPL with Packages > Julia > Open Console.

Note: in Juno, the “Auto-hide Menu Bar” option can turn off the File menu. This can be disorienting. If you find this disabled and the “Settings” window is unavailable, search for the Auto-Hide setting by typing `Ctrl + Shift + P`, then typing `Auto-hide` in the search bar. Ensure the option is de-selected. Alternatively, hit the `Alt` key to toggle the menu bar.

1.1.8 Packages and Modules

Julia utilizes packages, modules, and functions. Packages are comprised of Modules, Modules are comprised of functions, and the concept of functions is similar to that in many other programming languages and will not be covered in depth in this section. See <https://docs.julialang.org/en/v1/manual/modules/index.html> for more information.

1.1.8.1 Installing Registered Modules

To install modules via the REPL, it is easiest to enter the `pkg` mode by typing the right bracket `]`. Then, using the `add` command, bring packages and modules into scope. This avoids using dot notation and extra typing.

```
julia> ]
pkg> add MLDataUtils
# backspace to exit Pkg mode
julia> using MLDataUtils
```

Now the `MLDataUtils` module is available in the project environment. Check the status of all packages installed in the current environment by entering `pkg` mode,

then typing `status`. This should list all available packages.

Create Modules

Modules are typically comprised of functions and other data structures. There are many packages and modules in active development and well-maintained by the Julia community, many of which are registered, official modules that can be accessed and used via the REPL. A complete list of official packages is available at the Julia Registry (<https://pkg.julialang.org/docs/>).

The syntax of a module is typically as follows:

```
#dir/julia-lib.jl

module myJuliaLib
function some_func()
    do_something
    return x
end
end
```

1.1.8.2 Custom Modules

Julia makes using not only official registered packages simple, but affords users the opportunity to create and utilize their own packages and modules using both `git` and local file structures. To develop custom modules and bring them into a project scope, create the module to be “exported” and modify the `LOAD_PATH` variable to include the filepath (or URL if hosted on `git`) where the module file is. In the example below, if the `myJuliaLib` module is in the `\path\to\app` directory, add the path into the environment variable as follows:

```
# first check the LOAD_PATH var:
julia> LOAD_PATH
3-element Array{String,1}:
 "@"
 "@v#.#"
 "@stdlib"

julia> push!(LOAD_PATH, "/path/to/app")
4-element Array{String,1}:
 "@"
 "@v#.#"
 "@stdlib"
 "/path/to/app"
```

Alternatively:

```
julia> push!(LOAD_PATH, https://github.com/mygitrepo.git)4-element Array{String,1}:
 "@"
 "@v#.#"
 "@stdlib"
 "mygitrepo.git"
```

1.2 DOCOPTS

Docopts in Julia are similar to Python, Ruby, and other languages.

1.2.1 Installation

Again, most of Julia's packages that are not in the base code are contained within git repositories. [4]. To get started with DocOpts in Julia, issue the following command within the Julia REPL:

```
Pkg.add("DocOpt")
```

Usage example is given from the following website: <https://github.com/docopt/DocOpt.jl> [5]. Create a file called `naval-fate.jl`:

```
#dir/naval-fate.jl
doc = """Naval Fate.
https://github.com/JuliaLang/METADATA.jl

Usage:
  naval_fate.jl ship new <name>...
  naval_fate.jl ship <name> move <x> <y> [--speed=<kn>]
  naval_fate.jl ship shoot <x> <y>
  naval_fate.jl mine (set|remove) <x> <y> [--moored|--drifting]
  naval_fate.jl -h | --help
  naval_fate.jl --version

Options:
  -h --help      Show this screen.
  --version      Show version.
  --speed=<kn>   Speed in knots [default: 10].
  --moored       Moored (anchored) mine.
  --drifting     Drifting mine.
```

In the same directory as `naval-fate.jl`, run the following in the Julia REPL:

```
julia> using DocOpt # import docopt function
julia> args = docopt(doc, version=v"2.0.0")
```

1.3 EXTERNAL PROGRAMS

Several implementations provide the ability to access Julia through external programs such as Python, C, and Java, and vice versa.

1.3.1 Resources

Pycall package by Steven G. Johnson <https://github.com/JuliaPy/PyCall.jl>.

1.3.2 Python

Julia and Python express similar syntax, and both are dynamically-typed. The Pycall package enables interoperability between the two languages, extending Julia functionality and reach while maintaining its desirable qualities. This package creates the ability to call Python functions and import Python modules from Julia, among other functions. [6].

1.3.2.1 Installation

Start Julia, and run `Pkg.add("Pycall")`.

1.3.2.2 Import Python Modules

To access Python with Julia, we use the Julia syntax `using Pycall`. Here is a simple example [6]:

```
julia> using Pycall
julia> math = pyimport("math")
julia> math.sin(math.pi / 4)
returns ≈ 1/√2 = 0.70710678...
```

This is in contrast to Julia's built in `sin` function, which some consider bulky and less efficient:

```
julia> sin(fill(1.0, (2,2)))
2×2 Array{Float64,2}:
 0.454649  0.454649
 0.454649  0.454649
```

Pycall in Virtualenvs

This course recommends using virtualenvs in Python. It is therefore important to note that Pycall “uses the virtualenv it was built with by default, even if you switch virtualenvs.” [6]. This applies to virtual environments created using `[venv]` and `[virtualenv]`. Python virtual environments created by conda are not currently supported. To continue interoperability with Julia while using a different virtualenv, Pycall recommends switching virtualenvs and running:

```
julia>rm(Pkg.dir("Pycall", "deps", "PYTHON")); Pkg.build("Pycall")`

activate virtual environment in system shell and start Julia
$ source PATH/TO/bin/activate
$ julia
```

```
julia> ENV["PYCALL_JL_RUNTIME_PYTHON"] = Sys.which("python")
"PATH/T0/bin/python3"

julia> using PyCall

julia> pyimport("sys").executable
"PATH/T0/bin/python3"
```

1.3.3 Java in Julia

Julia interacts with Java through the use of the `JavaCall.jl` package. [7] “Static and instance method with primitive or object arguments and return values are callable.” [7].

1.3.3.1 Installation

```
julia> Pkg.add("JavaCall")
```

1.3.3.2 Usage

```
julia> using JavaCall
julia> JavaCall.init(["-Xmx128M"])
julia> jlm = @jimport java.lang.Math
JavaObject{:java.lang.Math} (constructor with 2 methods)

julia> jcall(jlm, "sin", jdouble, (jdouble,), pi/2) 1.0
julia> jnu = @jimport java.net.URL
JavaObject{:java.net.URL} (constructor with 2 methods)

julia> gurl = jnu((JString,), "http://www.google.com")
JavaObject{:java.net.URL}{Ptr{Void} @0x0000000108ae2aa8}

julia> jcall(gurl, "getHost", JString, ())
"www.google.com"

julia> j_u_arrays = @jimport java.util.Arrays
JavaObject{:java.util.Arrays} (constructor with 2 methods)

julia> jcall(j_u_arrays, "binarySearch", jint, (Array{jint,1}, jint),
           [10,20,30,40,50,60], 40)
```

1.4 PARALLEL COMPUTING

Though not intended as a replacement for Hadoop, parallelism in Julia is fairly straight-forward [2].

1.4.1 Coroutines or Tasks

Julia is optimized for high performance computing in a distributed environment. One method of parallelism in Julia is constructed via Coroutines which use

communication primitives to communicate between processes. The Channel function in Julia makes passing variables, results, and objects between tasks possible [1]. To create a Channel of size(int) we pass:

```
julia> c1 = Channel{32}
```

We can also specify a `Type{T}` before the size, though if not specified, the Channel can hold any type object. Type declaration in Julia is important. The Julia documentation [1] provides a simple example of creating results in one Channel and taking them into a second Channel.

```
julia> c1 = Channel{Int}(32)
julia> c2 = Channel{Tuple}(32)

# and a function `foo` which reads items from c1, processes the item read
# and writes a result to c2,

julia> function foo()
    while true
        data = take!(c1) # process data
        put!(c2, result) # write out result
    end
end

# we can schedule `n` instances of `foo` to be active concurrently.
julia> for i in 1:3
    @async foo()
end
```

The `put!` method appends an item to a Channel, unless the Channel is full in which case it blocks until a `take!` is issued against that Channel.

The documentation provides are more complex example which we develop here. This application creates the asynchronous channels “jobs” and “results”, creating a job in one and storing the results in the second, along with a simulated amount of time.

```
# first create the channels of type {T} and size (int).
julia> const jobs = Channel{Int}(32);
julia> const results = Channel{Tuple}(32);

julia> function do_work()
    for job_id in jobs
        exec_time = rand()
        sleep(exec_time)
        put!(results, (job_id, exec_time))
    end
end;

julia> function make_jobs(n)
    for i in 1:n
        put!(jobs, i)
    end
end;

julia> n=12

julia> @async make_jobs(n)
```



```
julia> for i in 1:4
    @async do_work()
end

julia> @elapsed while n > 0
    job_id, exec_time = take!(results)
    println("$job_id finished in $(round(exec_time; digits=2)) seconds")
    global n = n-1
end
```

1.4.2 Multi-threading

This section is under development.

1.4.3 Green-threading

This section is under development.

1.4.4 Hadoop and Julia

While development and interactivity between Spark, Hadoop, and Julia is also optimized for high performance computing in a distributed environment. While development and interactivity between Spark, Hadoop, and Julia is robust, the documentation is limited. The Elly.jl package [8] is a Hadoop and Yarn client for Julia.

1.5 AWS IN JULIA

There are two Julia libraries that provide programmatic interface with AWS: AWS.jl and AWSCore.jl. We will focus on AWSCore.jl, as it is under active development and provides more infrastructure types and high-level packages. We include a note about AWS.jl for completeness. The following information can be found at <https://juliacloud.github.io/AWSCore.jl/build/index.html> [9]

1.5.1 AWSCore.jl

AWSCore.jl (hereinafter AWSCore) was only successfully tested on Julia 0.6.0. Note that multiple versions of Julia can be installed and run by linking the binary files to the shell session while using bash and the Julia REPL. See “Installing Julia” for more information.

1.5.1.1 Credentials

Most `AWSCore` functions take an `AWSConfig` dictionary as an initial argument, which passes in both credentials and region variables. Credentials can be gathered either from the default `~/.aws/credentials` file or by setting them as environment variables as discussed elsewhere in this chapter.

`AWSCore` will attempt to read in credentials from (in order) Environment Variables, an `~/.aws/credentials` file, or EC2 Instance Credentials which can be set via the AWS CLI. *Note that the third method is not recommended!*

Here, we use an AWS Credentials file located `~/.aws/credentials` (Ubuntu). Ensure that the user has appropriate permissions to create, delete, and modify AWS objects and services. We begin with a simple script that will allow creation of S3 containers. To set up a script that will create an S3 container and store a simple object, we initialize the Julia Dev environment. First, create a new directory called “Julia App” and populate with one `.jl` file:

```
> mkdir JuliaApp
> cd JuliaApp
> touch julia-app.jl
```

Assuming Julia is properly installed, simply type the command for the executable created earlier. For instance, `0.6`. This will start a Julia REPL inside the `JuliaApp` directory just created. Open a text/file editor to edit the `julia-app.jl` file:

```
$ emacs julia-app.jl
```

Julia is comparable to Python in its import statements in that to use a particular package or library, the user must specify that package prior to its use. With the blank `.jl` open, import and initialize the following packages:

```
julia> Pkg.add("AWSCore")
julia> Pkg.add("AWSS3")
julia> using AWSS3
julia> using AWSCore.Services
```

Now include in the script the `AWSConfig()` argument, which typically contains AWS Credentials and a Region parameter:

```
julia> aws = AWSCore.aws_config()
```

Before proceeding, save the file and run the script in the opened Julia REPL. Make sure that the working directory is the same as where your script is located by issuing in Julia `pwd()`. To run the script, type in the Julia REPL:

```
julia> import Pkg
julia> include("julia-app.jl")
```

Read AWS credentials into your environment session by adding the following command to your script:

```
julia> aws = aws_config()
```

1.5.1.2 S3 Containers

You should see some output indicating that the credentials have been created or read successfully. Now we can create a bucket and put an object in the bucket. Very basically, the syntax for creating a bucket using the low-level AWSCore services is: `s3_create_bucket(credentials, bucketname)`. Add the following to your script, ensuring that your name complies with AWS bucket naming conventions:

```
julia> s3_create_bucket(aws, "<bucket.name>")
julia> s3_enable_versioning(aws, "<bucket.name>")
```

1.5.1.2.1 Listing buckets and Objects

Now we can run our script and verify that the bucket was created (see running Julia script above). We will further develop this script to include docopts to make managing S3 buckets and objects much easier. To expand on the previous example, add in the `s3_list_buckets(credentials)` OR `list_objects(credentials, bucket)` functions.

1.5.1.2.2 Adding objects to existing bucket

```
julia> s3_put(aws, "<bucket.name>", "key", "Hello!")
```

1.5.2 AWS.jl

Julia's AWS API (AWS.jl) is straightforward to configure and is similar to Python. Using the Julia-AWS API carries the advantage of users not having to switch between environments and languages. The API functionality in Julia is currently limited, however, to EC2, S3, SQS, and Auto Scaling. <https://github.com/JuliaCloud/AWS.jl>. Advanced APIs have not been tested yet.

Further development is in progress with the AWSSDK.jl package.

1.5.3 REST in julia

The basic structure for creating a GET REST function in Julia is as follows. This can be typed directly in the REPL:

```
using HTTP
function make_API_call(url)
    try
        response = HTTP.get(url)
        return String(response.body)
    catch e
        return "Error occurred : $e"
    end
end

response = make_API_call("http://jsonplaceholder.typicode.com/users")
println(response)
```

This example can be extended to include POST, PUT, and DELETE functions quickly. For more a robust web framework, we turn to Genie. <https://github.com/GenieFramework/Genie.jl>. There are several other web frameworks for Julia, including Mux.jl (<https://github.com/JuliaWeb/Mux.jl>), and HTTP.jl (<https://github.com/JuliaWeb/HTTP.jl>).

1.5.3.1 Genie.jl

“Genie is a full-stack MVC web framework that provides a streamlined and efficient workflow for developing modern web applications. It builds on Julia’s strengths (high-level, high-performance, dynamic, JIT compiled), exposing a rich API and a powerful toolset for productive web development.” <https://github.com/GenieFramework/Genie.jl> Genie actively supports Julia 1.0 and above.

1.5.3.1.1 Import Genie

To install Genie, in the Julia REPL in Atom IDE, issue the following command:

```
julia> import Pkg; Pkg.add("Genie")
```

This step only needs to be completed once per environment.

1.5.3.1.2 API Backend

Extending functionality for the app is similar to development in Django and Rails. See https://genieframework.github.io/Genie.jl/guides/Simple_API_backend.html for more detailed information.

In the example below, we create the API backend to retrieve data through REPL interaction with Genie. In the REPL, we create and edit a new file called for this example `rest.jl`. *Note:* The current working directory can be checked with `pwd()` and changed with `cd("path\\to\\files")`. In the Atom editor, add the following to a new file named `rest.jl`:

```
#rest.jl
julia> using Genie
julia> import Genie.Router: route
julia> import Genie.Renderer: json

julia> Genie.config.run_as_server = true

julia> route("/") do
    (:message => "Hi there!") |> json
end

julia> Genie.startup()
```

Now the server can be started and accessed at <https://127.0.0.1:8000> via the REPL:

```
julia> include("rest.jl")
```

Now we have the template for a running server, which can be expanded. The output of the code below will GET a JSON object. In the `rest.jl` file, add the following code.

```
#rest.jl
route("/echo", method = POST) do
    message = jsonpayload()
    (:echo => (message["message"] * " " ^ message["repeat"]) |> json)
end

route("/send") do
    response = HTTP.request("POST", "http://localhost:8000/echo", [{"Content-Type", "application/json"}],
        """{"message":"hello", "repeat":3}""")

    response.body |> String |> json
end

Genie.AppServer.startup(async = false)
```

Save the above code above in the current project directory. Then from the REPL or IDE REPL, run the script again:

```
julia> include("rest.jl")
```

1.5.3.1.3 Adding Parameters to the URL

We can also pass in parameters via URL. For instance, a string to filter or pass into a function.

```
#rest.jl
Base.convert(::Type{Float64}, s::SubString{String}) = parse(Float64, s)
route("/somefloats/:x::Float64/:y::Float64") do
    "x+y is $(Genie.Router.@params(:x) + Genie.Router.@params(:y))"
end
```

Then start the server by running the script, and request <http://localhost:8000/somefloats?x=2.5&y=4.5> in a browser.

1.5.3.1.4 Create an app

Genie provides scalability and ease of management through a web framework similar to Rails and Django. To create the framework for a new app, issue the following commands in the REPL:

```
julia> mkdir("directory-name")
julia> cd("directory-name")
julia> using Genie
julia> Genie.newapp("TestGenieApp")
```

Verify the app was created by typing <https://127.0.0.1:8000> in a browser. The Genie welcome mat should appear. To shut down the server, type `ctrl + c`. To restart the app via the IDE REPL, type:

```
julia> startup()
```

1.5.3.1.5 Train Test Split Now we can extend functionality further by

creating a Genie app. The app will be an API service including a GET implementation, which will call a Julia library designed to split data into train and test datasets.

Adding routes and functionality can be done either via the REPL or by editing the `routes.jl` file directly. For instance, the `somefloats()` function above could be added directly into the `routes.jl` file as in the above example.

Additionally, we can add a `lib` directory into the root directory and include pre-written code or modules in the app. In the following examples, Julia will be used to retrieve data from a website, then split that data into train and test sets.

1.5.3.1.6 Getting, Shuffling, Splitting data:

Syntax in Julia for gathering and manipulating data is similar to that in Python. For the purposes of this text, we will download and process the well-known Iris dataset using the RDatasets package, then split the dataset into train and test sets. For a complete list of the data available in Rdatasets, visit the following website: <https://github.com/JuliaStats/RDatasets.jl>. To begin, bring the RDatasets into scope by adding the following lines of code to your `routes.jl` file.

```
#routes.jl
julia> using RDatasets
```

Next, define the frame of a simple function as follows: *Note* there is no colon after the parameter parenthesis as in Python.

```
# routes.jl
route("/getdata") do
    data = data = dataset("datasets", "iris")
    data = shuffleobs(data)
    train, test = splitobs(data, at = 0.7, obsdim=1)
    return train, test
end
```

After making this fundamental change to the `routes.jl` file in the project directory, the environment must be re-activated. Ensure the project directory is the current working directory. In Atom IDE, this can be set by selecting in the menu: Julia -> Working Directory -> Select. Then, access the `pkg` mode by typing the right bracket `]`.

```
# enter package mode by typing right bracket
julia>]
#activate the environment for the current working directory:
pkg> activate .

#backspace to get back to Julia REPL
julia> using Genie
julia> Genie.loadapp()
julia> startup()
```

As the app is further tested and modified, it can be stopped using `ctrl + c`, then started again using `startup()`. The response in the browser window should look like the entire unformatted Iris dataset. Formatting, and utilizing Genie's Model-View-Controller architecture is beyond the scope of this chapter.

1.5.3.1.7 Extending functionality

The application structure as is works with basic functionality, but could become unwieldy very quickly. To scale the app one step further, we can place custom

modules and/or packages into the `/lib` directory. Genie will then add the custom content in the `lib` filepath into the `LOAD_PATH` environment variable recursively and automatically load dependencies. Add the filepath:

```
julia> mkdir("lib")
```

The directory should appear in the project environment. Add a file called `myJuliaLib.jl`. Add the functionality previously developed, along with the `module` syntax to export the module to the rest of the environment. This can be extended and modified to include many functions and data structures.

```
#myJuliaLib.jl
module myJuliaLib

using RDatasets
using MLDataUtils

function getdata()
    data = dataset("datasets", "iris")
    data = shuffleobs(data)
    train, test = splitobs(data, at=.7)
    return data
end
end
```

Finally, modify the `routes.jl` to call the custom content. Remove the actual function from the `/getdata` route, and replace it with a call to the custom `getdata()` function in `myJuliaLib`.

```
#routes.jl
route("/getdata") do
    myJuliaLib.getdata()
end
```

The entire `routes.jl` file should appear as follows:

```
using Genie.Router
using RDatasets
using myJuliaLib
using MLDataUtils

route("/") do
    serve_static_file("welcome.html")
end

route("/error500") do
    error_500("Something went wrong")
end

route("/error404") do
    error_404("the page you want")
end

route("/getdata") do
    myJuliaLib.getdata()
end
```

Reload the environment, restart the app, and type <http://127.0.0.1:8000/getdata>

into a browser window for the response. To continue development, utilize the Model-View-Controller approach to application development within Genie.

1.6 NETWORK

This section is still under development.

1.7 OPTIONAL TYPING

This section is still under development.

2 SCALA

2.1 SCALA FOR CLOUD COMPUTING

Scala is a multi-paradigm programming language aiming to integrate the features of object oriented paradigm with that of functional programming. Scala is a statically typed language. Scala and Java are interoperable in the sense that libraries written in either language can be used in Scala or Java.

2.1.1 Language

2.1.1.1 Install Scala

On macOS you can use Homebrew as follows:

```
brew update  
brew install scala
```

For instructions on how to install Scala on other platforms please visit the Scala website at <https://www.scala-lang.org/download/>.

2.1.1.2 Hello World! in Scala

Open a terminal and run Scala:

```
scala
```

If Scala is installed on your machine you should get the following message:

```
Welcome to Scala 2.12.8 (Java HotSpot(TM) 64-Bit Server VM, Java 11.0.2).  
Type in expressions for evaluation. Or try :help.
```

```
scala>
```

Now you can write and execute codes, for example:

```
println("hello world from scala!")
```

Now copy the line above into a text file and name it helloworld.scala. From the terminal run the following command:

```
scala helloworld.scala
```

The result of running this command is the same as executing the code in repl.

2.1.1.3 Basics

To apply a method on a expression, Scala uses dot notation:

```
"hello!".size
```

The code above applies the method `size` on the expression “hello!” and returns the following (the output is an instance of `Int` named ‘res0’ with a value of 6):

```
res0: Int = 6
```

As another example, consider the method `until` in the class `Int`:

```
def until(end: Int): collection.immutable.Range
```

This method returns a range from the object that calls the method up to but not included the value of the `end` parameter. Here is how we can apply this method to a number, let’s say 7:

```
7.until(15)
```

Scala allows the infix syntax and hence you can also use the following syntax to achieve the same outcome:

```
7 until 15
```

operators are methods which are usually used with the infix syntax: the following codes are equivalent:

```
4 + 5
```

```
4.+(5)
```

Methods are defined using the keyword `def`:

```
def trapezoidArea (base1 : Double, base2 : Double, height : Double): Double =  
(base1 + base2) / 2 * height
```

The definition above defines a function with three input parameters of type `Double` and it returns a double value.

We could use a block to define the body of the method above:

```
def trapezoidArea (base1 : Double, base2 : Double, height : Double) : Double =  
{(base1 + base2)/ 2 * height}
```

A block is defined by braces `{...}`. The last statement of a block determines its value. Anything you define inside a block is accessible only inside the block.

For more on the basic features of Scala language refer to the following resources: * [Scala Interactive Excersices](#) * [Tour of Scala](#)

2.1.1.4 Classes

Let us define a class called person:

```
class Person(var firstName : String, var lastName : String){  
  
  override def toString : String = s"$firstName $lastName"  
  
}  
  
val fred = new Person("Fred", "Fredian")  
println(fred)
```

Save the above script in a file named LearnTheLanguage.scala and run the code as the following:

```
scala LearnTheLanguage.scala
```

The class `Person` has two fields and one method. The primary constructor is in the class signature. A class has one primary constructor (this is the constructor that you define in the signature of the class) and any (including none) number of auxiliary constructors. An auxiliary constructor is defined using the keyword `this`. All auxiliary constructors must start with a call to a preceding auxiliary constructor or the primary constructor.

In the class `Person` above, we override the method `toString`. The method `toString` is a method in the class `AnyRef`. The class `AnyRef` is the supertype of any reference type. `AnyRef` is the equivalent of `java.lang.Object`.

[Figure 1](#) (taken from [10]) shows a subset of Scala type hierarchy:

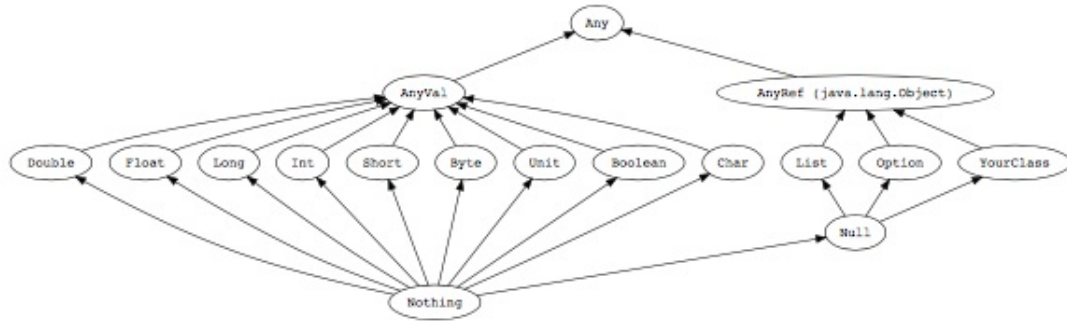


Figure 1: A subset of the type hierarchy

The top class `Any` has methods like `hashCode`, `isInstanceOf`, `asInstanceOf`, `toString` and the methods for equality (more on the class is available at <https://www.scala-lang.org/api/2.12.1/scala/Any.html>). The class `AnyVal` does not add any methods. All value types are derived from `AnyVal`. The class `AnyRef` adds some more methods: `wait`, `notify` and `synchronized` and `eq`. The method `eq` checks whether two references refer to the same object. If you want to compare two objects of a class by their values, you should override the `equals`. For example for the class `Person` defined above we should override `equals` as follows:

```
final override def equals(other: Any) = {
  val otherPerson = other.asInstanceOf[Person]
  if (otherPerson == null) false
  else firstName == otherPerson.firstName && lastName == otherPerson.lastName
}
```

Note that the method `equals` returns a boolean. In Scala, the type of the last statement in the body a method, is the return type of the method. Here we could make it implicit by declaring the return type as follows:

```
final override def equals(other: Any) : Boolean = {
  val otherPerson = other.asInstanceOf[Person]
  if (otherPerson == null) false
  else firstName == otherPerson.firstName && lastName == otherPerson.lastName
}
```

But Scala has type inference capability and can infer the types wherever there are enough information for it to infer types.

In addition to a top class `Any`, Scala has also a bottom class `Nothing`. The class `Nothing` is a subtype of any other type. Scala also a `Null` type. The class `Null` has one and only instance called `null`. The class `Null` is a subtype of all reference types and therefore `null` is assignable to any reference type but not to value types.

Members of a class are `public` by default. You can use the access modifier `private` to limit access to the members. A `private` member can be accessed only within the class. In the example above, the members `firstName` and `lastName` are public because they have been defined using `var`. In the primary constructor, any parameter that is defined using either `var` or `val` is public. Otherwise it is `private`. `var` indicates that the thing to be defined is mutable while `val` indicates that the thing to be defined is immutable.

In Java we have the concept of `interfaces` where an `interface` contains only the signature of some methods. `Traits` in Scala are similar to interfaces in Java. However unlike interfaces in Java, a trait can contain attributes and code. Like Java, in Scala a class can have only one supertype. In case you want to define a class with more than one supertype, you can use `traits`. Traits can not be instantiated but classes and objects (an `object` in Scala, which is defined by the keyword `object` is just a singleton class) can extend traits.

Subtyping in Scala is similar to that of Java. Let us modify our code and add the entity `Employee` as a subclass of `Person`:

```
class Person(val firstName : String, val lastName : String){
  override def toString : String = s"$firstName $lastName"
}

class Employee(firstName:String, lastName:String, val employeeId : String) extends Person(firstName, lastName){
  var salary = 0.0
}

val fred = new Employee("Fred", "Fredian", "410")
println(fred)
```

Note that in the code above the primary constructor of `Employee` is calling the primary constructor of its supertype. In Scala, only the primary constructor can call a superclass constructor.

Like Java, you can use the keyword `abstract` to define a class that cannot be instantiated. In an abstract class, you can have methods or members that are not defined (that is, you can define a method without defining its body).

2.1.1.4.1 Pattern matching

Pattern matching means that for a given data we have a sequence of patterns against which we match the data. Java `switch` statements are examples of pattern matching.

Assume you have two different jobs: selling apples and software programming:

```
sealed trait Profession
case class SellingApples (buy : Double, sell : Double, logistics : Double) extends Profession
case class Programming (hours : Double, perHourRate : Double, cost : Double) extends Profession
```

In this example we used the keyword `sealed`. This keyword can be used with a class or a trait. A `sealed class` cannot be inherited directly except when the inheriting class (or trait) is defined in the same source file. Similarly a `sealed trait` cannot be extended except in the same source file that contains the definition of the trait. One of the usages of `sealed traits` is when we want to have an alternative to Java's `enums`.

The differences between `class` and `case class` are the followings:

- In order to instantiate an object of a class we need to use the keyword `new` whereas this is not needed for `case classes`.
- Two instances, `a` and `b` of a `class` are not equal, that is `a == b` will return false. Whereas two instances `a` and `b` of a `case class` when they have the same values for their members are equal, that is `a == b` will return true.

Now we can use the pattern matching feature of Scala as the following:

```
def calculateProfit(profession : Profession) : Double =
  profession match {
    case SellingApples (buy, sell, logistics) => sell - (buy + logistics)
    case Programming (hours, perHourRate, cost) => (hours * perHourRate) - cost
  }
```

The `match` keyword, first checks whether the profession is `SellingApples` and if it is so, it extracts the parameters and then evaluates the expression of the right hand side of the arrow, `=>`. If the match is not successful it proceeds to the next case, if any.

The example above is an instance of an `algebraic data type`. An algebraic data type is a sealed trait together with several case classes that extend the sealed trait. Whenever you have an `is-a` relationship between the concepts in the domain, it may be a good idea to model the domain using an algebraic data type.

2.1.1.5 Scala as a functional language

Functions in Scala are first-class values and therefore you can pass them to other

methods or a method can return another method as its return value. A function that one of its input or output parameters is itself a function, is called a `higher order function`.

Consider the following example:

```
def cost(area : (Double , Double , Double) => Double,
x : Double, y : Double, z : Double, costPerUnit : Double) : Double =
area(x, y, z) * costPerUnit
```

Now we can call this method as the following:

```
cost(trapezoidArea, 2, 3, 4, 10)
```

Similarly we could pass an `anonymous function` to `cost`:

```
cost((x, y, z) => (x + y)/2 * z, 2, 3, 4, 10)
```

2.1.1.6 Scala as a tool to create Domain Specific Languages(DSL)

Scala was designed so that it would support creating DSLs ([refer to here for more](#)) . Among the features and capabilities that are in particular helpful for creating DSLs are the followings[11]:

... implicit function definitions, which allow values of one type to be lifted to values of a different type, the ability to call methods on objects without dots and parentheses, case classes, partial functions, call-by-name, user-defined operators composed of symbols, operator overloading, as well as other features such as higher-order functions and lambda expressions.

2.1.1.6.1 Cloud computing with Scala and GridGain

[GridGain](#) a company that provides Grid and Cloud computing platforms and services, is using a DSL written in Scala that uses GridGain's cloud computing development systems to create cloud computing platforms. Here is a presentation which includes simple practical programs using the DSL in Scala:

[Cloud Computing with Scala and GridGain](#)

2.1.2 Parallel programming in Scala

2.1.2.1 Parallel collections

Let us start with a simple example on how to run a simple operation in parallel. In this example we want to transform a list of strings to all-lowercase. Here is the sequential version of the code:

```
scala> val x = List("Scala", "SPARK", "PAR", "ParelleL")
scala> x.map(_.toLowerCase)
```

The method `map`, applies the function that it receives as its argument to each member of the list.

Scala has a parallel counterpart to a number of its important sequential collections. For example `ParArray`, `mutable.ParHashMap` and `immutable.ParHashMap`. Scala has a special method `par` that when you call this method on a sequential collection, it will copy all the elements in that collection to an equivalent parallel collection. So to run the above example in parallel we only need to apply the function `par` as the following:

```
scala> val x = List("Scala", "SPARK", "PAR", "ParelleL").par
scala> x.map(_.toLowerCase)
```

2.1.2.2 Actor model

To do concurrent and parallel programming in Java, we usually use threads. Creating threads and managing their intercommunication is complex as you need to handle how they use shared resources, and how they communicate to each other. Especially you often need to use locks and monitors to prevent deadlocks. Another paradigm in concurrent programming is the `actor model`. In this model we have actors instead of threads. An actor is simply an object that can send and receive messages to other actors. Therefore it is an event-based mechanism.

Scala had an [Actors](#) library but starting from Scala 2.11.0 that library is deprecated in favor of [Akka](#). Akka is a toolkit written in Scala for developing concurrent event-driven applications.

Let us develop a simple concurrent program using Akka. We develop this program using [sbt](#) build tool. We also use IntelliJ Community Edition as our IDE. To install IntelliJ Community Edition and its Scala plugin please follow [this tutorial](#). When you have the IDE and the Scala plugin installed then follow

the instructions in [the second tutorial](#) to build a sample project (call it `scala_example`) using the IDE and sbt build tool.

Now in your `scala_example` project replace the content of the file `build.sbt` with the following:

```
name := "scala_example"

version := "0.1"

scalaVersion := "2.12.8"

libraryDependencies ++= Seq(
  "com.typesafe.akka" %% "akka-actor" % "2.5.21",
  "com.typesafe.akka" %% "akka-testkit" % "2.5.21" % Test
)
```

This file tells the sbt build tool that it needs to download the Akka related libraries for this project.

Next replace the content of the file `Main.scala` with the following:

```
package scala_example

import akka.actor.{Actor, ActorRef, ActorSystem, Props}

object Person {
  def props(name: String): Props = Props(new Person(name))

  case class MoneySent(amount: Double)

  case class MoneyRequested(amount: Double)
}

class Person(name: String) extends Actor {
  import Person._

  def receive = {

    case MoneySent(amount) => {
      println(name + " sent $" + amount)
    }
    case MoneyRequested(amount) => {
      println(name + " requested $" + amount)
    }
  }
}

object Main extends App {

  import Person._

  val system: ActorSystem = ActorSystem("helloAkka")
  val fred: ActorRef = system.actorOf(Person.props("fred"))
  val andi = system.actorOf(Person.props("andi"))
  fred ! new MoneyRequested(100)
  andi ! new MoneySent(100)
}
```

Now run the program using the run command of sbt. The output of the program will be either of the followings:

this:

```
[info] Running scala_example.Main
andi sent $100.0
fred requested $100.0
```

or this:

```
[info] Running scala_example.Main
fred requested $100.0
andi sent $100.0
```

In the example we defined a class named `Person` which extends the trait `Actor`. This class overrides the `receive` method of the `Actor` trait to handle the messages that it receives. What happens when an actor receives a message, is the following: * The actor maintains a queue for the messages that it receives. So when the actor receives a message, it adds the message to its queue. * The actor executes the message that is in front of its queue.

Note that the internal state of an actor can be changed only through messages. Now an actor processes messages one at a time, so there will be no races regarding the internal states of the actors and therefore there is no need to have locks or monitors. Also senders are not locked and as soon as they sent a message they can continue doing other things.

3 DEVTOOLS

3.1 REFCARDS



Learning Objectives

- Obtain quickly information about technical aspects with the help of reference cards.
-
-

We present you with a list of useful short reference cards. This cards can be extremely useful to remind yourself about some important commands and features. Having them could simplify your interaction with the systems, We not only collected here some refcards about Linux, but also about other useful tools and services.

If you like to add new topics, let us know via your contribution (see the contribution section).

CheatSheets

- [CheatSheets](#)

Editors

- [Emacs](#)
- [Vi](#)
- [Vim](#)

Documentation

- [LaTeX](#)
- [RST](#)

Linux

- [Linux](#)
- [Makefile](#)
- [Git](#)

Cloud/Virtualization

- [Openstack](#)
- [Openstack](#)
- [vagrant](#)

SQL

- [SQL](#)

Languages

- [R](#)

Python

- [Python](#)
- [PythonData](#)
- [Numpy/Pandas](#)
- [PythonTutorial](#)
- [Python](#)
- [Python](#)
- [PythonAPIIndex](#)
- [Python3](#)

3.2 VIRTUAL BOX

For development purposes we recommend that you use for this class an Ubuntu virtual machine that you set up with the help of virtualbox. We recommend that you use the current version of ubuntu and do not install or reuse a version that you have set up years ago.

As access to cloud resources requires some basic knowledge of linux and security we will restrict access to our cloud services to those that have

demonstrated responsible use on their own computers. Naturally as it is your own computer you must make sure you follow proper security. We have seen in the past students carelessly working with virtual machines and introducing security vulnerabilities on our clouds just because “it was not their computer.” Hence, we will allow using of cloud resources only if you have demonstrated that you responsibly use a linux virtual machine on your own computer. Only after you have successfully used ubuntu in a virtual machine you will be allowed to use virtual machines on clouds.

A *cloud drivers license test* will be conducted. Only after you pass it we will let you gain access to the cloud infrastructure. We will announce this test. Before you have not passed the test, you will not be able to use the clouds. Furthermore, you do not have to ask us for join requests to cloud projects before you have not passed the test. Please be patient. Only students enrolled in the class can get access to the cloud.

If you however have access to other clouds yourself you are welcome to use the, However, be reminded that projects need to be reproducible, on our cloud. This will require you to make sure a TA can replicate it.

Let us now focus on using virtual box.

3.2.1 Installation

First you will need to install virtualbox. It is easy to install and details can be found at

- <https://www.virtualbox.org/wiki/Downloads>

After you have installed virtualbox you also need to use an image. For this class we will be using ubuntu Desktop 16.04 which you can find at:

- <http://www.ubuntu.com/download/desktop>

Please note some hardware you may have may be too old or has too little resources to be useful. We have heard from students that the following is a minimal setup for the desktop machine:

- multi core processor or better allowing to run hypervisors
- 8 GB system memory
- 50 GB of free hard drive space

For virtual machines you may need multiple, while the minimal configuration may not work for all cases.

As configuration we often use

minimal

1 core, 2GB Memory, 5 GB disk

latex

2 core, 4GB Memory, 25 GB disk

A video to showcase such an install is available at:



[Using Ubuntu in Virtualbox \(8:08\)](#)



Please note that the video shows the version 16.04. You should however use the newest version which at this time is 18.04.

If you specify your machine too small you will not be able to install the development environment. Gregor used on his machine 8GB RAM and 25GB disk space.

Please let us know the smallest configuration that works.

3.2.2 Guest additions

The virtual guest additions allow you to easily do the following tasks:

- Resize the windows of the vm

- Copy and paste content between the Guest operating system and the host operating system windows.

This way you can use many native programs on you host and copy contents easily into for example a terminal or an editor that you run in the Vm.

A video is located at



[Virtualbox \(4:46\)](#)

Please reboot the machine after installation and configuration.

On OSX you can once you have enabled bidirectional copying in the Device tab with

OSX to Vbox:

command c shift CONTRL v

Vbox to OSX:

shift CONTRL v shift CONTRL v

On Windows the key combination is naturally different. Please consult your windows manual. If you let us know TAs will add the information here.

3.2.3 Exercises

E.Virtualbox.1:

Install ubuntu desktop on your computer with guest additions.

E.Virtualbox.2:

Make sure you know how to paste and copy between your host and guest operating system.

E.Virtualbox.3:

Install the programs defined by the development configuration.

E.Virtualbox.4:

Provide us with the key combination to copy and paste between Windows and VBox.

3.3 VAGRANT



Learning Objectives

- Be able to experiment with virtual machines on your computer before you go on a cloud.
 - Simulate a virtual cluster with multiple VMs running on your computer if it is big enough.
-
-

A convenient tool to interface with Virtual Box is vagrant. Vagrant allows us to manage virtual machines directly from the commandline. It support also other providers and can be used to start virtual machines and even containers. The latest version of vagrant includes the ability to automatically fetch a virtual machine image and start it on your local computer. It assumes that you have virtual box installed. Some key concepts and advertisement are located at

- <https://www.vagrantup.com/intro/index.html>:

Detailed documentation for it is located

- <https://www.vagrantup.com/docs/index.html>

A list of *boxes* is available from

- <https://app.vagrantup.com/boxes/search>

One image we will typically use is Ubuntu 18.04. Please note that older version may not be suitable for class and we will not support any questions about them. This image is located at

- <https://app.vagrantup.com/ubuntu/boxes/bionic64>

3.3.1 Installation

Vagrant is easy to install. You can go to the download page and download and install the appropriate version:

- <https://www.vagrantup.com/downloads.html>

3.3.1.1 macOS

On MacOS, download the dmg image, and click on it. You will find a pkg in it that you double click. After installation vagrant is installed in

- `/usr/local/bin/vagrant`

Make sure `/usr/local/bin` is in your `PATH` Start a new terminal to verify this.

Check it with

```
echo $PATH
```

If it is not in the path put

```
export PATH=/usr/local/bin:$PATH
```

in the terminal command or in your `~/.bash_profile`

3.3.1.2 Windows



 students contribute

3.3.1.3 Linux



 students contribute

3.3.2 Usage

To download, start and login into install the 18.04:

```
host$ vagrant init ubuntu/bionic64
host$ vagrant up
host$ vagrant ssh
```

Once you are logged in you can test the version of python with

```
vagrant@ubuntu-bionic:~$ sudo apt-get update
vagrant@ubuntu-bionic:~$ python3 --version
Python 3.6.5
```

To install a newer version of python, and pip you can use

```
vagrant@ubuntu-bionic:~$ sudo apt-get install python3.7
vagrant@ubuntu-bionic:~$ sudo apt-get install python3-pip
```

To install the light weight idle development environment in case you do not want o use pyCharm, please use

```
vagrant@ubuntu-bionic:~$ sudo apt-get install idle-python
```

So that you do not have to always use the number 3, you can also set an alias with

```
alias python=python3
```

When you exit the virtual machine with the

```
exit command
```

It does not terminate the VM. You can use from your host system the commands such as

```
host$ vagrant status
host$ vagrant destroy
host$ vagrant suspend
host$ vagrant resume
```

to manage the vm.

3.4 PACKER



No

Packer is an open source tool for creating identical machine images for multiple platforms from a single source configuration. Packer runs on every major operating system, and creates machine images for multiple platforms in parallel form configuration specifications.

Some key concepts are located at

- <https://www.packer.io/intro/index.html>

Detailed documentation is located at

- <https://www.packer.io/docs/index.html>

Use cases for packer is located at

- <https://www.packer.io/intro/use-cases.html>

3.4.1 Installation

Installation instructions for all platforms is located at

- <https://www.packer.io/intro/getting-started/install.html>

3.4.2 Usage

In the Section [vagrant](#) we use vagrant to start up an Ubuntu 18.04 virtual machine. Once the VM was up and running, vagrant allowed the user to log in and setup the VM according to the user's requirements. In that example, the user ran commands to install and upgrade software dependencies:

1. upgrade from Python 3.6.5 to Python 3.7
2. installing python3-pip and idle-python
3. alias `python` to `python3`

Let us assume that the VM is now in a desirable state for the purpose of doing development on a large number of virtual machines and you want to distribute it to the rest of your team or community so that all are using the same environment. You could simply send your team members a copy of your Ubuntu 18.04 VirtualBox VM assuming they will be developing on VMs using VirtualBox. However, let us assume one community member wants to develop on Google Cloud Platform, another on AWS and another on OpenStack. In this case, they will each need to figure out how to import a VirtualBox VM into the respective cloud vendor they're utilizing. Packer can help this situation by codifying the state of the development environment with a single configuration file which can then be used to create images in different cloud environments.

Assuming packer has been installed, let's create a packer JSON file that will build an Ubuntu 18.04 image and provision it as we did manually using Vagrant. In this example, we will create the image in Google Compute Platform.

First download your Google Cloud credentials according to the documentation at

- <https://www.packer.io/docs/builders/googlecompute.html#running-without-a-compute-engine-service-account>

Save the credential file as `accounts.json`. Also, determine the project ID you will use in your Google Cloud Platform account. In this example, we will use `my_project_id` for our project ID.

Next save the following JSON to a file named `e516.json`:

```
{
  "variables": {
    "google_project_id": null
  },
  "builders": [
    {
      "type": "googlecompute",
      "account_file": "account.json",
      "project_id": "{{ user `google_project_id` }}",
      "image_name": "ubuntu-1804-dev-e516",
      "source_image": "ubuntu-1804-bionic-v20180911",
      "ssh_username": "packer",
      "zone": "us-central1-a"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "expect_disconnect": true,
      "inline": [
        "sudo apt-get update -y",
        "sudo apt-get install -y python3.7 python3-pip idle-python3.7",
        "echo `alias python='python3'` > .bash_aliases"
      ]
    }
  ]
}
```

```
]
}
```

The packer file format specifies 3 sections, `variables`, `builders` and `provisioners`. The `variables` section allows you to declare variables that are to be used in the rest of the document. By declaring a variable in this section, for example `google_project_id`, it allows the user to pass in the value of that variable via the packer command line.

The `builders` section allows you to declare the builders for any cloud vendor supported by packer. The list of supported vendors can be found here:

- <https://www.packer.io/docs/builders/index.html>

In our example, we define the builder for Google Cloud Platform which requires our credential file (account.json), our project ID, base image name, ssh username and zone.

Finally, the `provisioners` section allows the user to customize the base image defined in the `builders` section. In our example, we simply use the `shell` provisioner which allows us to type in shell commands to provision the image as we want it. Here we install `python3.7`, `python3-pip` and `idle-python3.7`. We also write out an aliases file so that upon login, the user can access `python3.7` using the `python` alias.

To build the image, we now run packer:

```
$ packer build -var 'google_project_id=my_project_id' e516.json
```

You will see output that shows the progress of packer as it starts up and provisions the instance. Upon success, packer will create an image from the instance and clean up after itself:

```
$ googlecompute output will be in this color.
```

```
==> googlecompute: Checking image does not exist...
==> googlecompute: Creating temporary SSH key for instance...
==> googlecompute: Using image: ubuntu-1804-bionic-v20180911
==> googlecompute: Creating instance...
    googlecompute: Loading zone: us-central1-a
    googlecompute: Loading machine type: n1-standard-1
    googlecompute: Requesting instance creation...
    googlecompute: Waiting for creation operation to complete...
    googlecompute: Instance has been created!
==> googlecompute: Waiting for the instance to become running...
    googlecompute: IP: 104.154.21.240
==> googlecompute: Waiting for SSH to become available...
==> googlecompute: Connected to SSH!
==> googlecompute: Provisioning with shell script: /var/folders/rm/g1h4bhf54x750jzjyryckmnc0001xd/T/packer-shell1210916201
    googlecompute: Get:1 http://archive.canonical.com/ubuntu bionic InRelease [10.2 kB]
...

```

```

googlecompute: Setting up idle-python3.7 (3.7.0-1~18.04) ...
googlecompute: Processing triggers for libc-bin (2.27-3ubuntu1) ...
googlecompute: Processing triggers for ureadahead (0.100.0-20) ...
googlecompute: Processing triggers for systemd (237-3ubuntu10.3) ...
==> googlecompute: Deleting instance...
googlecompute: Instance has been deleted!
==> googlecompute: Creating image...
==> googlecompute: Deleting disk...
googlecompute: Disk has been deleted!
Build 'googlecompute' finished.

==> Builds finished. The artifacts of successful builds are:
--> googlecompute: A disk image was created: ubuntu-1804-dev-e516

```

You can now click on the list of images in the Google Compute Platform console to see your new image. The new image is ready to use for development.

Next, let's add a builder for an AWS AMI. Before we do that, setup your AWS credentials using the AWS CLI according to the documentation here:

- <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html>

Ensure your `default` profile is saved under `~/.aws/credentials`.

Update the `e516.json` so that the contents is as follows:

```

{
  "variables": {
    "google_project_id": null,
    "image_name": "ubuntu-1804-dev-e516",
    "ssh_username": "packer"
  },
  "builders": [
    {
      "type": "googlecompute",
      "account_file": "account.json",
      "ssh_username": "{{ user `ssh_username` }}",
      "project_id": "{{ user `google_project_id` }}",
      "image_name": "{{ user `image_name` }}",
      "source_image": "ubuntu-1804-bionic-v20180911",
      "zone": "us-central1-a"
    },
    {
      "type": "amazon-efs",
      "ssh_username": "{{ user `ssh_username` }}",
      "profile": "default",
      "ami_name": "{{ user `image_name` }}",
      "source_ami": "ami-0bbe6b35405ecebdb",
      "instance_type": "t2.micro",
      "region": "us-west-2"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "expect_disconnect": true,
      "inline": [
        "sudo apt-get update -y",
        "sudo apt-get install -y python3.7 python3-pip idle-python3.7",
        "echo `alias python='python3'` > .bash_aliases"
      ]
    }
  ]
}

```

Note that we've added the AWS builder in the `builders` section and that we've refactored the `ssh_username` and `image_name` to the `variables` section since those variable hold values that can be reused in both the Google Compute and AWS builders.

Let us rerun packer:

```
packer build -var 'google_project_id=my_project_id' e516.json
```

You will see output that states the image already exists in your Google Compute account and so packer smartly skips building that image. The output also shows the progress of packer as it starts up and provisions the instance in AWS. Upon success, packer will create an AMI from the instance and clean up after itself:

```
amazon-eks output will be in this color.
googlecompute output will be in this color.

==> googlecompute: Checking image does not exist...
==> amazon-eks: Prevalidating AMI Name: ubuntu-1804-dev-e516
==> googlecompute: Image ubuntu-1804-dev-e516 already exists.
==> googlecompute: Use the force flag to delete it prior to building.
Build 'googlecompute' errored: Image ubuntu-1804-dev-e516 already exists.
Use the force flag to delete it prior to building.
amazon-eks: Found Image ID: ami-0bbe6b35405ecebdb
==> amazon-eks: Creating temporary keypair:
packer_5bad9d99-f631-1778-1e83-afd19ad0d5cc
==> amazon-eks: Creating temporary security group for this instance:
packer_5bad9d9b-38c5-252d-0368-74aa75bfb286
==> amazon-eks: Authorizing access to port 22 from 0.0.0.0/0
in the temporary security group...
==> amazon-eks: Launching a source AWS instance...
==> amazon-eks: Adding tags to source instance
amazon-eks: Adding tag: "Name": "Packer Builder"
amazon-eks: Instance ID: i-0d0383f9f84b54051
```

You can now click on the list of images in the AWS EC2 console to see your new AMI. The new AMI is ready to use for development.

3.5 UBUNTU ON AN USB STICK

In case you cannot install any programs on your development computer most often the easiest way is to use the hardware but boot the OS from a USB stick. Make sure you have access to the Bios or your system to actually boot from a USB device before you start this activity.

3.5.1 Ubuntu on an USB stick for macOS via Command Line

The easiest way to create an ubuntu distribution that can be booted from an USB stick is done via command line. The original Web page for this method is available at this [\[link\]](#).

We have copied some of the information from this Web page but made enhancements to it. Currently all images are copied from that Web page.



Please test it out and improve if it does not work.

Our goal is to create a USB stick that has either Ubuntu 18.04 LTS that can be downloaded from this [\[link\]](#). You will need a USB stick/flash drive. We recommend a 8GB or larger. Please let us know if it works for you on larger than 8GB drives.

We assume that you downloaded the iso from ubuntu to a folder called */iso*. Next we open a terminal and *cd* into the folder */iso*. Now we need to convert the iso to an image file. This is done as follows and you need to execute the command for the version of ubuntu you like to use.

Your folder will look something like this

```
$ ls -l
ubuntu-18.04-desktop-amd64.iso
```

You will need to generate an image with the following command

```
$ hdiutil convert ubuntu-18.04-desktop-amd64.iso -format UDRW -o ubuntu-18.04-desktop-amd64.img
```

macOS will append a *.dmg* behind the name. At this time **do not** plug in your usb stick. Just issue the command

```
$ diskutil list
```

Observe the output. Now plug in the USB stick. Wait till the USB stick registers in the Finder. If this does not work find a new USB stick or format it. Execute the command

```
$ diskutil list
```

and observe the output again. Another device will register and you will see something like

```
/dev/disk2 (external, physical):
#:      TYPE NAME           SIZE      IDENTIFIER
0:      FDisk_partition_scheme *8.2 GB    disk2
1:      DOS_FAT_32 NO NAME     8.2 GB    disk2s1
```

Please note in this example the device path and number is recognized as

```
/dev/disk2
```

It also says external, which is a good sign as the USB stick is external. Next, we need to unmount the device with

```
$ diskutil unmountDisk /dev/diskN
```

where you replace the number N with the disk number that you found for the device. In our example it would be 2. If you see the error “Unmount of diskN failed: at least one volume could not be unmounted”, start Disk Utility.app and unmount the volume (do not eject). If it was successful, you will see

```
Unmount of all volumes on disk2 was successful
```

The next step is dangerous and you need to make sure you follow it. So please do not copy and paste, but read first, reflect and only if you understand it execute it. We know we say this all the time, but better saying it again instead of you destroying your system. This command also requires sudo access so you will either have to be in the sudo group, or use

```
$ su <your administrator name>
```

login and then execute the command under root.

```
$ sudo dd if=ubuntu-18.04-desktop-amd64.img.dmg of=/dev/diskN bs=1m
```

(Not tested: Using /dev/rdisk instead of /dev/disk may be faster according to the ubuntu documentation)

Ubuntu’s Web page also gives the following tips:

- “If you see the error dd: Invalid number ‘1m’, you are using GNU dd. Use the same command but replace bs=1m with bs=1M.”
- “If you see the error dd: /dev/diskN: Resource busy, make sure the disk is not in use. Start Disk Utility.app and unmount the volume (do not eject).”

You will see an error window popping up telling you: **The disk inserted was not readable by this compute.** Please, leave the window as is and instead type in on the terminal.

```
$ diskutil eject /dev/diskN
```

Now remove the flash drive, and press in the error window **Ignore**

Now you have a flash drive with ubuntu installed and you can boot from it. To do so, please

restart your Mac and press option key

while the Mac is restarting to choose the USB-Stick

You will need a plug for USB keyboard, USB mouse, and network cable.

There are some issue from this point on.

```
$ sudo apt-get update
```

Add universe to the window for application updates

see <https://help.ubuntu.com/community/Repositories/Ubuntu>

```
$ sudo apt-get install vnc4server
```

Start the server and set up a password

```
$ vncserver
```

The next section is untested and needs verification.

3.5.1.1 Boot from the USB Stick

To boot from the USB stick, you need to restart or power-on the Mac with the USB stick inserted while you press the Option/alt key.

The launch *Startup Manager* will be started showing a list of bootable devices connected to the machine. Your USB stick should appear as gold/yellow and labelled *EFI Boot*. Use your cursor keys to move to the most right EFI boot device in that list (likely the USB stick) and press ENTER. YOU can also use the mouse.

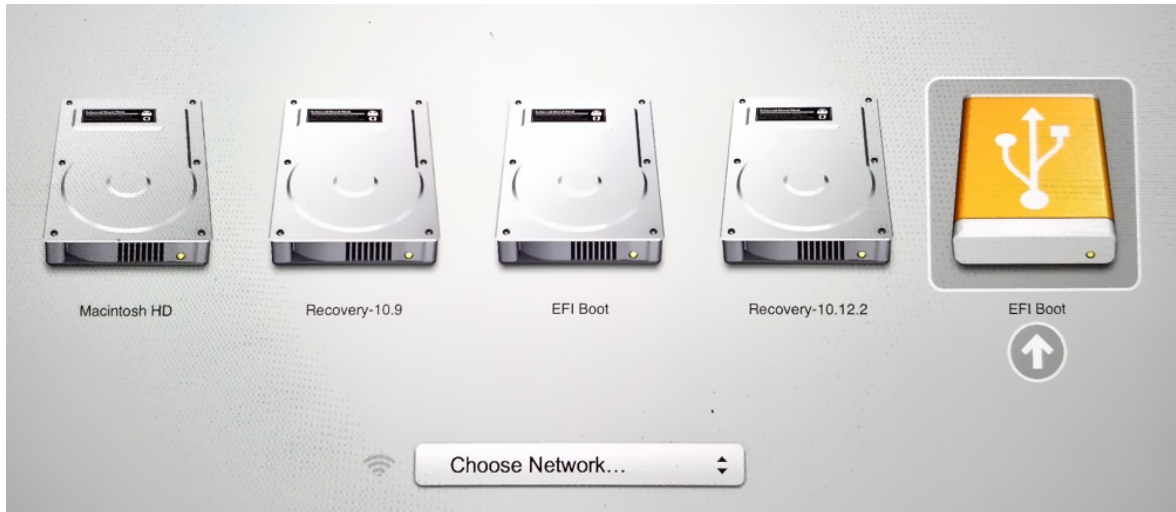


Figure: Boot Screen

A boot menu will shortly start up and after you press again ENTER your machine will boot into Ubuntu.

For more information on how to setup ubuntu see:

- <https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0>

After you have booted and logged in, you need to update the distribution. We recommend that you switch on Universe in the applications settings.

Next you need to issue in the command terminal

```
$ sudo apt-get update
```

You will likely see some warnings with number 95 which you can ignore. Please report your experience and we update this page based on your feedback.

3.5.2 Ubuntu on an USB stick for macOS via GUI

An alternative to the Command Line solution to create an USB stick with bootable Ubuntu on is to use the macOS GUI. This method is more complex than the command line solution. In addition as we are learning about cloud computing in this book, it is of advantage to learn how to do this from commandline as the replication of the approach via commandline is easier and more scalable. However for completeness, we have also included here the GUI-based method.

The material in this section was copied and modified from

- <https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-macos>

You will need a USB stick/flash drive. We recommend a 8GB or larger. Please let us know if it works for you on larger than 8GB drives.

3.5.2.1 Install Etcher

Etcher is a tool that allows you to easily write an ISO onto a USB stick. Etcher is integrated in the macOS GUI environment and allows to drag the iso into it for burning. Etcher can be found at

- <https://etcher.io/>

As this is an application from unidentified developers (not registered in the apple store), you need to enable it after downloading. To do so, you can enable the *App Store and identified developers* in the *Security and Privacy* pane in the System Preferences. IN case you get a warning about running the application, click *Open Anyway* in the same pane.

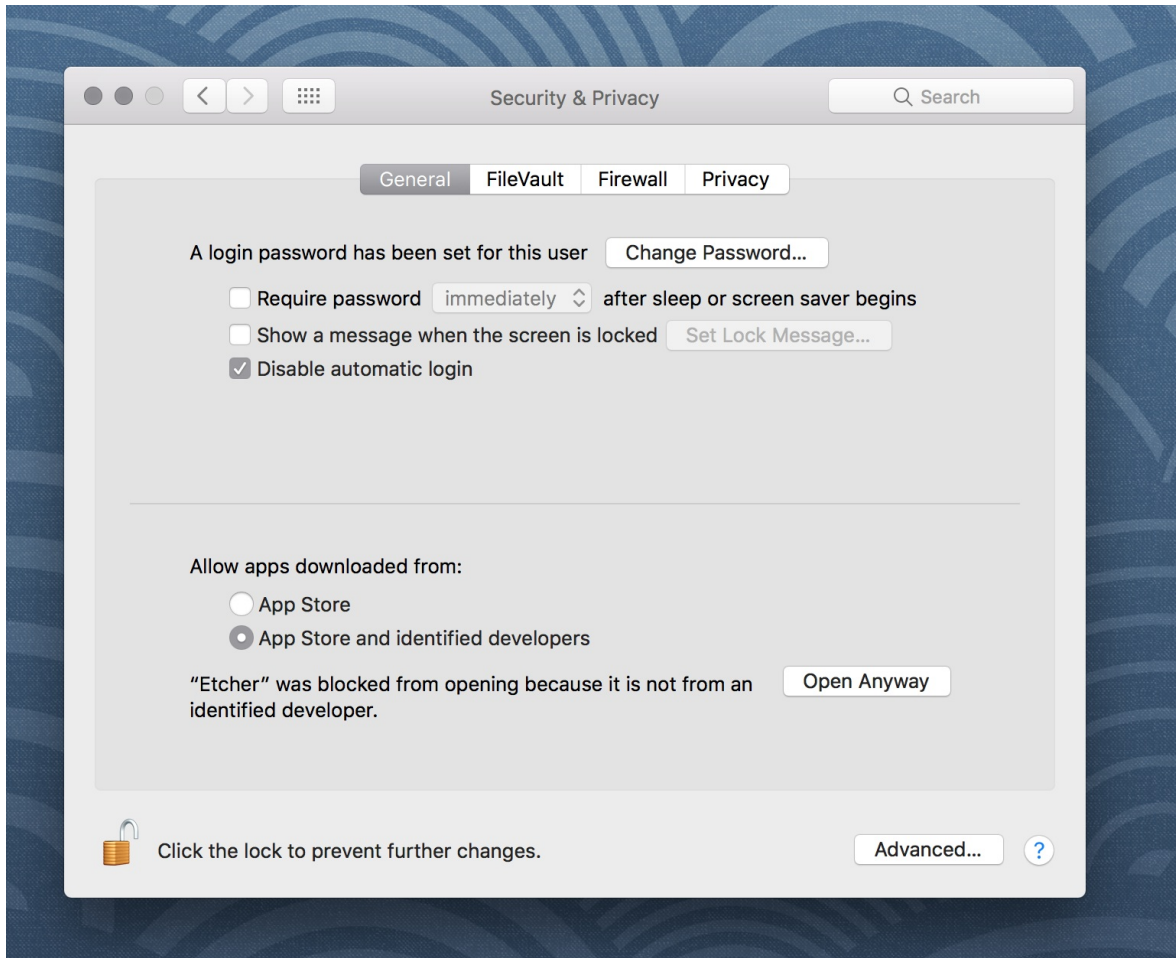


Figure: Setting

3.5.2.2 Prepare the USB stick

The Disk Utility needs to be used with caution as selecting the wrong device or partition can result in data loss.

Next you need to conduct the following steps which we copied from the Ubuntu Web page:

- Launch Disk Utility from Applications>Utilities or Spotlight search
- Insert your USB stick and observe the new device added to Disk Utility
- Select the USB stick device and select Erase from the tool bar (or right-click menu)
- Set the format to MS-DOS (FAT) and the scheme to GUID Partition Map
Check you've chosen the correct device and click Erase

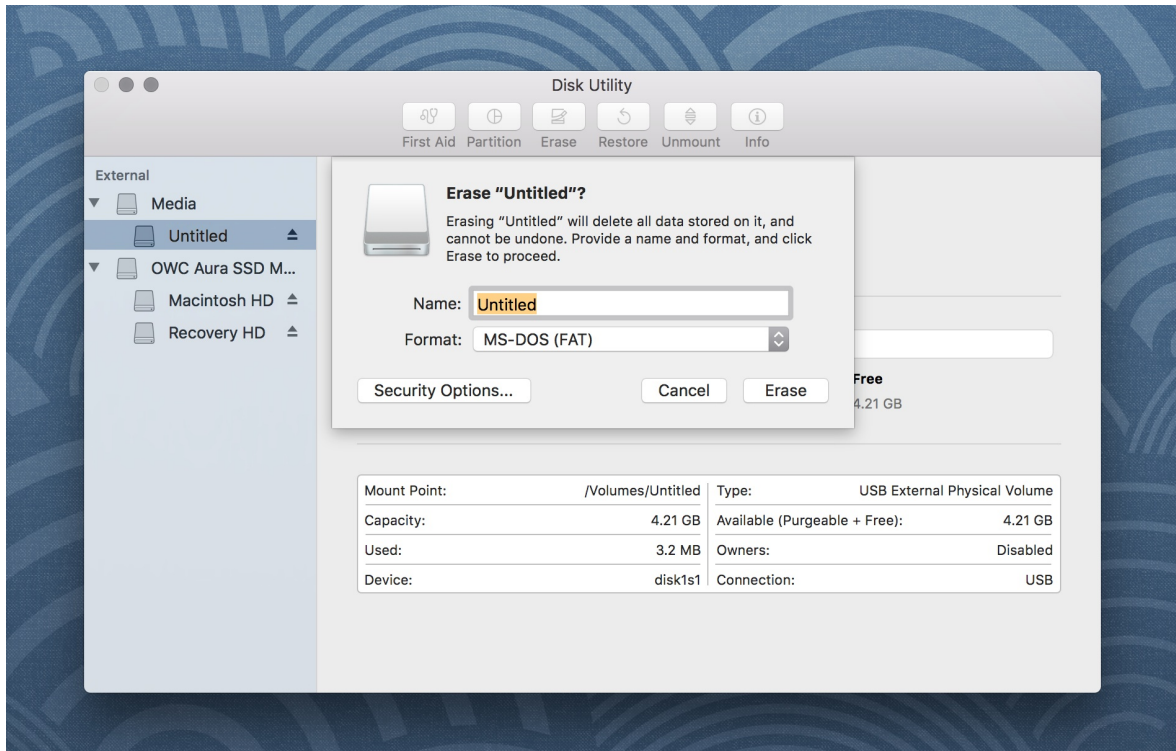


Figure: Diskutil

3.5.2.3 Etcher configuration

Next we use Etcher to configure and write to your USB device as follows (copied from the Ubuntu Web page):

- Select image will open a file requester from which should navigate to and select the ISO file downloaded previously. By default, the ISO file will be in your Downloads folder.
- Select drive, replaced by the name of your USB device if one is already attached, lets you select your target device. You will be warned if the storage space is too small for your selected ISO.
- Flash! will activate when both the image and the drive have been selected. As with Disk Utility, Etcher needs low-level access to your storage hardware and will ask for your password after selection.

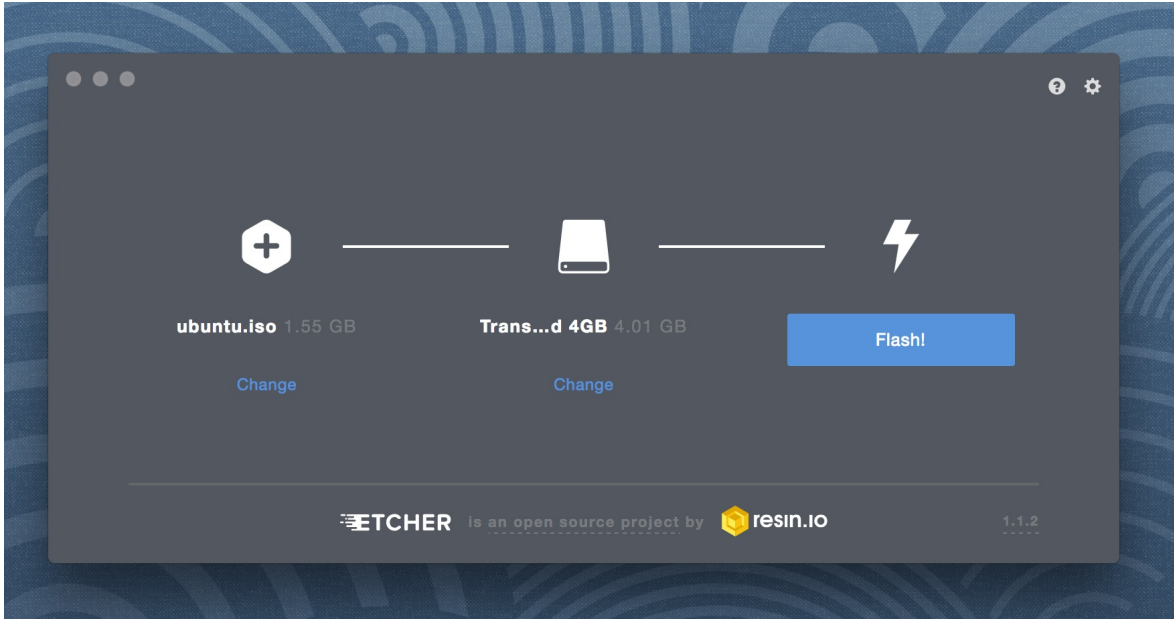


Figure: Etcher complete message

3.5.2.4 Write to the USB stick

When writing to the USB, Etcher will ask you for your password. It will write the ISO file, once you confirmed the password.

You will see the progress reported to the Etcher window. Once it has finished, Etcher will report on the successful process.

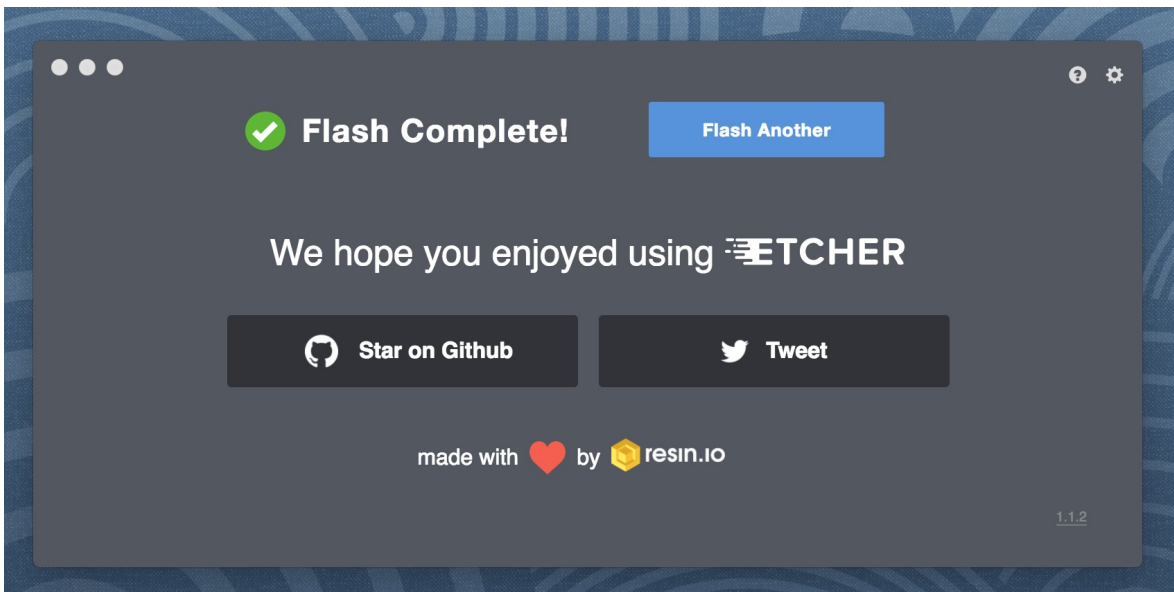


Figure: Etcher

After the write process has completed, macOS may inform you that *The disk you inserted was not readable by this computer*. Do not select Initialise. Instead, select Eject and remove the USB device.

3.5.3 Ubuntu on an USB stick for Windows 10

See exercise Development.Server.1

Material for this directions were taken from a detailed tutorial [\[link\]](#)

First you will need to install Rufus, which is a free program to create bootable USB drives on windows. Rufus is available at

- <https://rufus.akeo.ie/>

Next you need to launch Rufus, insert the USB stick, and observe that it is added to Rufus. Select the Device on which you like to place ubuntu. Be careful that you do not by accident use a wrong device.

Select the partition scheme and target system type set as MBR partition scheme for UEFI. (in case you have older hardware try MBR Partition Scheme for BIOS or UEFI instead).

Select the ubuntu iso file.

Next press the `start` button so we activate the write process. This will take quite a while. Select `Write in ISO Image mode (Recommended)`

Once the process is completed, try booting from it. How to activate the boot in your system depends on your hardware and vendor. Please consult with your documentation.

3.5.4 Exercise

Development.Server.1

If you are in need to but from a USB stick in Windows, please verify and expand on our section similar to the one provided by macOS. It does not matter if you chose a GUI or a commandline option via

`gitbash.`

3.6 GITHUB

3.6.1 Github



Learning Objectives

- Be able to use the github cloud services to collaborately develop contents and programs.
 - Be able to use github as part of an open source project.
-
-

In some classes the material may be openly shared in code repositories. This includes class material, papers and project. Hence, we need some mechanism to share content with a large number of students.

First, we like to introduce you to git and github.com (Section [1.1](#)). Next, we provide you with the basic commands to interact with git from the commandline (Section [1.12](#)). Then we will introduce you how you can contribute to this set of documentations with pull requests.

3.6.1.1 Overview

Github is a code repository that allows the development of code and documents with many contributors in a distributed fashion. There are many good tutorials about github. Some of them can be found on the github Web page. An interactive tutorial is for example available at

- <https://try.github.io/>

However, although these tutorials are helpful in many cases they do not address some cases. For example, you have already a repository set up by your organization and you do not have to completely initialize it. Thus do not just replicate the commands in the tutorial, or the once we present here before not evaluating their consequences. In general make sure you verify if the command

does what you expect **before** you execute it.

A more extensive list of tutorials can be found at

- <https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github>

The github foundation has a number of excellent videos about git. If you are unfamiliar with git and you like to watch videos in addition to reading the documentation we recommend these videos

- <https://www.youtube.com/user/GitHubGuides/videos>

Next, we introduce some important concepts used in github.

3.6.1.2 Upload Key

Before you can work with a repository in an easy fashion you need to upload a public key in order to access your repository. Naturally, you need to generate a key first which is explained in the section about ssh key generation (📺 [TODO: lessons-ssh-generate-key](#) include link) before you upload one. Copy the contents of your `.ssh/id_rsa.pub` file and add them to [your github keys](#).

More information on this topic can be found on the [github Web page](#).

3.6.1.3 Fork

Forking is the first step to contributing to projects on GitHub. Forking allows you to copy a repository and work on it under your own account. Next, creating a branch, making some changes, and offering a pull request to the original repository, rounds out your contribution to the open source project.



[Git 1:41 Fork](#)

3.6.1.4 Rebase

When you start editing your project, you diverge from the original version.

During your developing, the original version may be updated, or other developers may have some of their branches implementing good features that you would like to include in your current work. That is when *Rebase* becomes useful. When you *Rebase* to certain points, could be a newer Master or other custom branch, consider you graft all your on-going work right to that point.

Rebase may fail, because some times it is impossible to achieve what we just described as conflicts may exist. For example, you and the to-be-rebased copy both edited some common text section. Once this happens, human intervention needs to take place to resolve the conflict.



[Git 4:20 Rebase](#)

3.6.1.5 Remote

Collaborating with others involves managing the remote repositories and pushing and pulling data to and from them when you need to share work. Managing remote repositories includes knowing how to add remote repositories, remove remotes that are no longer valid, manage various remote branches and define them as being tracked or not, and more.

Though out this semester, you will typically work on two *remote* repos. One is the office class repo, and another is the repo you forked from the class repo. The class repo is used as the centralized, authority and final version of all student submissions. The repo under your own Github account is for your personal storage. To show progress on a weekly basis you need to commit your changes on a weekly basis. However make sure that things in the master branch are working. If not, just use another branch to conduct your changes and merge at a later time. We like you to call your development branch `dev`.

- <https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes>

3.6.1.6 Pull Request


Pull requests are a means of starting a conversation about a proposed change back into a project. We will be taking a look at the strength of conversation, integration options for fuller information about a change, and cleanup strategy

for when a pull request is finished.

 [Git 4:26 Pull Request](#)


3.6.1.7 Branch

Branches are an excellent way to not only work safely on features or experiments, but they are also the key element in creating Pull Requests on GitHub. Lets take a look at why we want branches, how to create and delete branches, and how to switch branches in this episode.

 [Git 2:25 Branch](#)


3.6.1.8 Checkout

Change where and what you are working on with the checkout command. Whether we are switching branches, wanting to look at the working tree at a specific commit in history, or discarding edits we want to throw away, all of these can be done with the checkout command.

 [Git 3:11 Checkout](#)

3.6.1.9 Merge

Once you know branches, merging that work into master is the natural next step. Find out how to merge branches, identify and clean up merge conflicts or avoid conflicts until a later date. Lastly, we will look at combining the merged feature branch into a single commit and cleaning up your feature branch after merges.

 [Git 3:11 Merge](#)

3.6.1.10 GUI

Using Graphical User Interfaces can supplement your use of the command line to get the best of both worlds. GitHub for Windows and GitHub for Mac allow

for switching to command line, ease of grabbing repositories from GitHub, and participating in a particular pull request. We will also see the auto-updating functionality helps us stay up to date with stable versions of Git on the command line.

 [Git 3:47 GUI](#)

There are many other git GUI tools available that directly integrate into your operating system finders, windows, ..., or PyCharm. It is up to you to identify such tools and see if they are useful for you. Most of the people we work with us git from the command line, even if they use PyCharm, eclipse, or other tools that have build in git support. You can identify a tool that works best for you.

3.6.1.11 Windows

This is a quick tour of GitHub for Windows. It offers GitHub newcomers a brief overview of what this feature-loaded version control tool and an equally powerful web application can do for developers, designers, and managers using Windows in both the open source and commercial software worlds. More: <http://windows.github.com>

 [Git 1:25 Windows](#)

3.6.1.12 Git from the Commandline

Although github.com provides a powerful GUI and other GUI tools are available to interface with github.com, the use of git from the commandline can often be faster and in many cases may be simpler.

Git commandline tools can be easily installed on a variety of operating systems including Linux, macOS, and Windows. Many great tutorials exist that will allow you to complete this task easily. We found the following two tutorials sufficient to get the task accomplished:

- <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
- <https://www.atlassian.com/git/tutorials/install-git>

Although the later is provided by an alternate repository to github. The installation instructions are very nice and are not impacted by it. Once you have installed git you need to configure it.

3.6.1.13 Configuration

Once you installed Git, you can need to configure it properly. This includes setting up your username, email address, line endings, and color, along with the settings' associated configuration scopes.



[Git 2:47 Configuration](#)

It is important that make sure that use the `git config` command to initialize git for the first time on each new computer system or virtual machine you use. This will ensure that you use on all resources the same name and e-mail so that git history and log will show consistently your checkins across all devices and computers you use. If you do not do this, your checkins in git do not show up in a consistent fashion as a single user. Thus on each computer execute the following commands:

```
$ git config --global user.name "Albert Zweistein"
$ git config --global user.email albert.zweistein@gmail.com
```

where you replace the information with the information related to you. You can set the editor to emacs with:

```
$ git config --global core.editor emacs
```

Naturally if you happen to want to use other editors you can configure them by specifying the command that starts them up. You will also need to decide if you want to push branches individually or all branches at the same time. It will be up to you to make what will work for you best. We found that the following seems to work best:

```
git config --global push.default matching
```

More information about a first time setup is documented at:

* <http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup>

To check your setup you can say:

```
$ git config --list
```

One problem we observed is that students often simply copy and paste instructions, but do not read carefully the error that is reported back and do not fix it. Overlooking the proper set of the `push.default` is often overlooked. Thus we remind you: **Please read the information on the screen when you set up.**

3.6.1.14 Upload your public key

Please upload your public key to the repository as documented in github, while going to your account and find it in settings. There you will find a panel SSH key that you can click on which brings you to the window allowing you to add a new key. If you have difficulties with this find a video from the github foundation that explains this.

3.6.1.15 Working with a directory that will be provided for you

In case your course provided you with a github directory, starting and working in it is going to be real simple. Please wait till an announcement to the class is send before you ask us questions about it.

If you are the only student working on this you still need to make sure that papers or programs you manage in the repository work and do not interfere with scripts that instructors may use to check your assignments. Thus it is god to still create a branch, work in the branch and than merge the branch into the master once you verified things work. After you merged you can push the content to the github repository.

Tip: Please use only **lowercase** characters in the directory names and no special characters such as `@ ; / _` and spaces. In general we recommend that you avoid using directory names with capital letters spaces and `_` in them. This will simplify your documentation efforts and make the URLs from git more readable. Also while on some OS's the directories *MyDirectory* is different from *mydirectory* on macOS it is considered the same and thus renaming from capital to lower case can not be done without first renaming it to another directory.

Your homework for submission should be organized according to folders in your clone repository. To submit a particular assignment, you must first add it using:


```
git add <name of the file you are adding>
```

Afterwards, commit it using:

```
git commit -m "message describing your submission"
```

Then push it to your remote repository using:

```
git push
```

If you want to modify your submission, you only need to:

```
git commit -m "message relating to updated file"
```

afterwards:

```
git push
```

If you lose any documents locally, you can retrieve them from your remote repository using:

```
git pull
```

3.6.1.16 README.yml and notebook.md

In case you take classes e516 and e616 with us you will have to create a README.yml and notebook.md file in the top most directory of your repository. It serves the purpose of identifying your submission for homework and information about yourself.

It is important to follow the format precisely. As it is yaml it is an easy homework to write a 4 line python script that validates if the README.yml file is valid. In addition you can use programs such as `yamllint` which is documented at

- <https://yamllint.readthedocs.io/en/latest/>

This file is used to integrate your assignments into a proceedings. An example is provided at

- <https://github.com/cloudmesh-community/hid-sample/blob/master/README.yml>

Any derivation from this format will not allow us to see your homework as our

automated scripts will use the README.yml to detect them. Make sure the file does not contain any TABs. Please also mind that all filenames of all homework and the main directory must be **lowercase** and do not include spaces. This will simplify your task of managing the files across different operating systems.

In case you work in a team, on a submission, the document will only be submitted in the author and hid that is listed first. All other readme files, will have for that particular artifact a `duplicate: yes` entry to indicate that this submission is managed elsewhere. The team will be responsible to manage their own pull requests, but if the team desires we can grant access for all members to a repository by a user. Please be aware that you must make sure you coordinate with your team.

We will not accept submission of homework as pdf documents or tar files. All assignments must be submitted as code and the reports in native latex and in github. We have a script that will automatically create the PDF and include it in a proceedings. There is no exception from this rule and all reports not compilable will be returned without review and if not submitted within the deadline receive a penalty.

Please check with your instructor on the format of the README.yaml file as it could be different for your class.

To see an example for the notebook.md file, you can visit our sample hid, and browse to the notebook.md file. Alternatively you can visit the following link

- <https://github.com/cloudmesh-community/hid-sample/blob/master/notebook.md>

The purpose of the notebook md file is to record what you did in the class to us. We will use this file at the end of the class to make sure you have recorded on a weekly basis what you did for the class. Inactivity is a valid response. Not updating the notebook, is not.

The sample directory contains other useful directories and samples, that you may want to investigate in more detail. One of the most important samples is the github issues (see Section [1.19](#)). There is even a video in that section about this and showcases you how to organize your tasks within this class, while copying

the assignments from piazza into one or more github issues. As we are about cloud computing, using the services offered by a prominent cloud computing service such as github is part of the learning experience of this course.

3.6.1.17 Contributing to the Document

It is relatively easy to contribute to the document if you understand how to use github. The first thing you will need to do is to create a fork of the repository. The easiest way to do this is to visit the URL

- <https://github.com/cloudmesh-community/book>

Towards the upper right corner you will find a link called **Fork**. Click on it and chose into which account you like to fork the original repository. Next you will create a colne from your corked directory. You will see in your fork a green clone button. You will see a URL that you can copy into your terminal. If the links does not include your username, it is the wrong link.

In your terminal you now say

```
git clone https://github.com/<yourusername>/book
```

Now cd into this directory and make your changes.

```
$ cd book
```

Use the usual git commands such as `git add`, `git commit`, `git push`

Note you will push into your local directory.

3.6.1.17.1 Stay up to date with the original repo

Form time to time you will see that others are contributing to the original repo. To stay up to date you want to not only sync from your local copy, but also from the original repo. To link your repo with what is called the upstream you need to do the following once, so you can issue `git pull` tha also pulls from the upstream

Make sure you have upstream repo defined:

```
$ git remote add upstream \  
https://github.com/cloudmesh-community/book
```

Now Get latest from upstream:

```
$ git rebase upstream/master
```

In this step, the conflicting file shows up (in my case it was refs.bib):

```
$ git status
```

should show the name of the conflicting file:

```
$ git diff <file name>
```

should show the actual differences. May be in some cases, It is easy to simply take latest version from upstream and reapply your changes.

So you can decide to checkout one version earlier of the specific file. At this stage, the re-base should be complete. So, you need to commit and push the changes to your fork:

```
$ git commit  
$ git rebase origin/master  
$ git push
```

Then reapply your changes to refs.bib - simply use the backed up version and use the editor to redo the changes.

At this stage, only refs.bib is changed:

```
$ git status
```

should show the changes only in refs.bib. Commit this change using:

```
$ git commit -a -m "new:usr: <message>"
```

And finally push the last committed change:

```
$ git push
```

The changes in the file to resolve merge conflict automatically goes to the original pull request and the pull request can be merged automatically.

You still have to issue the pull request from the Github Web page so it is registered with the upstream repository.

3.6.1.17.2 Resources

- [Pro Git book](#)
- [Official tutorial](#)
- [Official documentation](#)
- [TutorialsPoint on git](#)
- [Try git online](#)
- [GitHub resources for learning git](#) Note: this is for github and not for gitlab. However as it is for gt the only thing you have to do is replace github, for gitlab.
- [Atlassian tutorials for git](#)

In addition the tutorials from atlassian are a good source. However remember that you may not use bitbucket as the repository, so ignore those tutorials. We found the following useful

- What is git: <https://www.atlassian.com/git/tutorials/what-is-git>
- Installing git: <https://www.atlassian.com/git/tutorials/install-git>
- git config: <https://www.atlassian.com/git/tutorials/setting-up-a-repository#git-config>
- git clone: <https://www.atlassian.com/git/tutorials/setting-up-a-repository#git-clone>
- saving changes: <https://www.atlassian.com/git/tutorials/saving-changes>
- collaborating with git: <https://www.atlassian.com/git/tutorials/syncing>

3.6.1.18 Exercises

E.Github.1:

How do you set your favorite editor as a default with github config

E.Github.2:

What is the difference between merge and rebase?

E.Github.3:

Assume you have made a change in your local fork, however other users have since committed to the master branch, how can you make sure your commit works off from the latest information in the master

branch?

E.Github.4:

Find a spelling error in the Web page or a contribution and create a pull request for it.

E.Gitlab.5:

Create a README.yml in your github account directory provided for you for class.

3.6.1.19 Github Issues



[Github 8:29 Issues](#)

When we work in teams or even if we work by ourselves, it is prudent to identify a system to coordinate your work. While conducting projects that use a variety of cloud services, it is important to have a system that enables us to have a cloud service that enables us to facilitate this coordination. Github provides such a feature through its *issue* service that is embedded in each repository.

Issues allow for the coordination of tasks, enhancements, bugs, as well as self defined labeled activities. Issues are shared within your team that has access to your repository. Furthermore, in an open source project the issues are visible to the community, allowing to easily communicate the status, as well as a roadmap to new features.

This enables the community to participate also in reporting of bugs. Using such a system transforms the development of software from the traditional closed shop development to a truly open source development encouraging contributions from others. Furthermore it is also used as bug tracker in which not only you, but the community can communicate bugs to the project.

A good resource for learning more about issues is provided at

- <https://guides.github.com/features/issues/>

3.6.1.19.1 Git Issue Features

A git issue has the following features:

title

– a short description of what the issue is about

description

a more detailed description. Descriptions allow also to conveniently add check-boxed todo's.

label

a color enhanced label that can be used to easily categorize the issue. You can define your own labels.

milestone

a milestone so you can identify categorical groups issues as well as their due date. You can for example group all tasks for a week in a milestone, or you could for example put all tasks for a topic such as developing a paper in a milestone and provide a deadline for it.

assignee

an assignee is the person that is responsible for making sure the task is executed or on track if a team works on it. Often projects allow only one assignee, but in certain cases it is useful to assign a group, and the group identifies if the task can be split up and assigns them through check-boxed todo's.

comments

allow anyone with access to provide feedback via comments.

3.6.1.19.2 Github Markdown

GitHub uses markdown which we introduce you in Section [\[S:markdown\]](#).

As github has its own flavor of markdown we however also point you to [this](#) as a reference. We like to mention the special enhancements fo github's markdown that integrate well to support project management.

3.6.1.19.2.1 Task lists

Taks lists can be added to any description or comment in github issues To create a task list you can add to any item `[]`. This includes a task to be done. To make it as complete simple change it to `[x]`. Whoever the great feature of tasks is that you do not even have to open the editor but you can simply check the task on and of via a mouse click. An example of a task list could be

Post Bios

```
* [x] Post bio on piazza
* [ ] Post bio on google docs
* [ ] Post bio on github
* [ ] \ (optional) integrate image in google docs bio
```

In case you need to use a `(` have at the beginning ot the task text, you need to escape it with a `\`

3.6.1.19.2.2 Team integration

A person or team on GitHub can be mentioned by typing the username proceeded by the `@` sign. When posting the text in the issue, it will trigger a notification to them and allow them to react to it. It is even possible to notify entire teams, which are described in more detail at

- <https://help.github.com/articles/about-teams/>

3.6.1.19.2.3 Referencing Issues and Pull requests

Each issue has a number. If you use the `#` followed by the issue number you can refer to it in the text which will also automatically include a hyperlink to the task. The same is valid for pull requests.

3.6.1.19.2.4 Emojis

Although github supports emojis such as 🍌 we do not use them typically in our class.

3.6.1.19.3 Notifications

Github allows you to set preferences on how you like to receive notifications. You can receive them either via e-mail or the Web. This is controlled by configuring it in *your settings*, where you can set the preferences for participating projects as well as projects you decide to watch. To access the notifications you can simply look at them in the *notification* screen. In this screen when you press the ? you will see a number of commands that allow you to control the notification when pressing on one of them.

3.6.1.19.4 cc

To carbon copy users in your issue text, simply use /cc followed by the @ sign and their github user name.

3.6.1.19.5 Interacting with issues

Github has the ability to search issues with a search query and a search language that you can find out more about it at

<https://guides.github.com/features/issues/#search>

A dashboard gives convenient overviews of the issues including a *pulse* that lists todo's status if you use them in the issue description.

3.6.1.20 Glossary

The Glossary is copied from

- <https://cdcvs.fnal.gov/redmine/projects/cet-is-public/wiki/GitTipsAndTricks#A-suggested-work-flow-for-distributed-projects-NoSY>

Add

put a file (or particular changes thereto) into the index ready for a commit

operation. Optional for modifications to tracked files; mandatory for hitherto un-tracked files.

Branch

a divergent change tree (eg a patch branch) which can be merged either wholesale or piecemeal with the master tree.

Commit

save the current state of the index and/or other specified files to the local repository.

Commit object

an object which contains the information about a particular revision, such as parents, committer, author, date and the tree object which corresponds to the top directory of the stored revision.

Fast-forward

an update operation consisting only of the application of a linear part of the change tree in sequence.

Fetch

update your local repository database (not your working area) with the latest changes from a remote.

HEAD

the latest state of the current branch.

Index

a collection of files with stat information, whose contents are stored as objects. The index is a stored version of your working tree. Files may be staged to an index prior to committing.

Master

the main branch: known as the trunk in other SCM systems.

Merge

join two trees. A commit is made if this is not a fast-forward operation (or one is requested explicitly).

Object

the unit of storage in git. It is uniquely identified by the SHA1 hash of its contents. Consequently, an object can not be changed.

Origin

the default remote, usually the source for the clone operation that created the local repository.

Pull

shorthand for a fetch followed by a merge (or rebase if `-rebase` option is

used).

Push

transfer the state of the current branch to a remote tracking branch. This must be a fast-forward operation (see merge).

Rebase

a merge-like operation in which the change tree is rewritten (see Rebasing below). Used to turn non-trivial merges into fast-forward operations.

Remote

another repository known to this one. If the local repository was created with “clone” then there is at least one remote, usually called, “origin.”

Stage

to add a file or selected changes therefrom to the index in preparation for a commit.

Stash

a stack onto which the current set of uncommitted changes can be put (eg in order to switch to or synchronize with another branch) as a patch for retrieval later. Also the act of putting changes onto this stack.

Tag

human-readable label for a particular state of the tree. Tags may be simple (in which case they are actually branches) or annotated (analogous to a CVS tag), with an associated SHA1 hash and message. Annotated tags are preferable in general.

Tracking branch

a branch on a remote which is the default source / sink for pull / push operations respectively for the current branch. For instance, origin/master is the tracking branch for the local master in a local repository.

Un-tracked

not known currently to git.

3.6.1.21 Example commands

To work in your local directory you can use the following commands. Please note that these commands do not upload your work to github, but only introduce version control within your local files.

The command list is copied from

- <https://cdcv.s.fnal.gov/redmine/projects/cet-is-public/wiki/GitTipsAndTricks#A-suggested-work-flow-for-distributed-projects-NoSY>

3.6.1.21.1 Local commands to version control your files

Obtain differences with

```
$ git status
```

Move files from one part of your directory tree to another:

```
$ git mv <old-path> <new-path>
```

Delete unwanted tracked files:

```
$ git rm <path>
```

Add un-tracked files:

```
$ git add <un-tracked-file>
```

Stage a modified file for commit:

```
$ git add <file>
```

Commit currently-staged files:

```
$ git commit -m <log-message>
```

Commit only specific files (regardless of what is staged):

```
$ git commit -m <log-message>
```

Commit all modified files:

```
$ git commit -a -m <log-message>
```

Un-stage a previously staged (but not yet committed) file:

```
$ git reset HEAD <file>
```

Get differences with respect to the committed (or staged) version of a file:

```
$ git diff <file>
```

Get differences between local file and committed version:

```
$ git diff --cached <file>
```

Create (but do not switch to) a new local branch based on the current branch:

```
$ git branch <new-branch>
```

Change to an existing local branch:

```
$ git checkout <branch>
```

Merge another branch into the current one:

```
$ git merge <branch>
```

3.6.1.21.2 Interacting with the remote

Get the current list of remotes (including URIs) with

```
$ git remote -v
```

Get the current list of defined branches with

```
$ git branch -a
```

Change to (creating if necessary) a local branch tracking an existing remote branch of the same name:

```
$ git checkout <branch>
```

Update your local repository ref database without altering the current working area:

```
$ git fetch <remote>
```

Update your current local branch with respect to your repository's current idea of a remote branch's status:

```
$ git merge <branch>
```

Pull remote ref information from all remotes and merge local branches with their remote tracking branches (if applicable):

```
$ git pull
```

Examine changes to the current local branch with respect to its tracking branch:

```
$ git cherry -v
```

Push changes to the remote tracking branch:

```
$ git push
```

Push all changes to all tracking branches:

```
$ git push --all
```

3.6.2 Git Pull Request

3.6.2.1 Introduction

Git pull requests allow developers to submit work or changes they have done to a repository, The developers can then check the changes that have been proposed in the pull request, discuss and make changes if needed. After the content off the pull request has been agreed upon it can be merged to the repository to add the information or changes in the pull request into the repository.

3.6.2.2 How to create a pull request

In this document we will see how we can create a pull request for the Cloudmesh technologies repo that is located at

- <https://github.com/cloudmesh/technologies>

However if you do pull request on other directories, you just have to replace the url with that of the repository you like to use. A common one four our classes is also

- <https://github.com/cloudmesh-community/book>

Which contains this book.

You can either create a pull request through a branch or through a fork. In this document we will be looking at how we can create a pull request through a fork.

3.6.2.3 Fork the original repository

First you need to create a fork of the original repository. A fork is your own copy of the repository to which you can make changes to. To fork the Cloudmesh technologies goto [Cloudmesh technologies repo](#) and click on the Fork button on the top right corner. Now you can notice that instead of `c.cloudmesh/technologies` the name of the repo says `YOURGITUSERNAME/technologies`, where `YOURGITUSERNAME` is indeed your github user name. That is because you are now in your own copy of the `c.cloudmesh/technologies` repository. In our case the user name will be `pulasthi`.

3.6.2.4 Clone your copy

Now that you have your fork created, we can go ahead and clone it into our machine. Instructions on how to clone a repository can be found in the Github documentation - [Cloning a repository](#). Make sure that you clone your version of the technologies repo.

3.6.2.5 Adding an upstream

Before we can start working on our copy of the git repo it is good to add an upstream (a link to the original repo) so that we can get all the latest changes in the original repository into our copy. Use the following commands to add an upstream to `c.cloudmesh/technologies`. First go into the folder which contains your git repo that you cloned and execute the following command.

```
$ git remote add upstream https://github.com/cloudmesh/technologies.git'
```

To make sure you have added it correctly execute the following command

```
$ git remote -v
```

You should see something similar to the following as the output

```
origin https://github.com/pulasthi/technologies.git (fetch)
origin https://github.com/pulasthi/technologies.git (push)
upstream https://github.com/cloudmesh/technologies.git (fetch)
upstream https://github.com/cloudmesh/technologies.git (push)
```

3.6.2.6 Making changes

Now you can make changes to your repo as with any normal git repository. However to make sure you have the latest copy from the original execute the following command before you start making changes. This will pull the latest changes from the original `cloudmesh/technologies` into your local copy

```
$ git pull upstream master
```

Now make the needed changes commit and push, the changes will be pushed to your copy of the repo i Github, not the `cloudmesh/technologies` repo.

3.6.2.7 Creating a pull request

Once we have changes pushed, you can go into your repository in Github to create a pull request. As seen in `@#fig:button-pullrequest`, you have an button named `pull request`

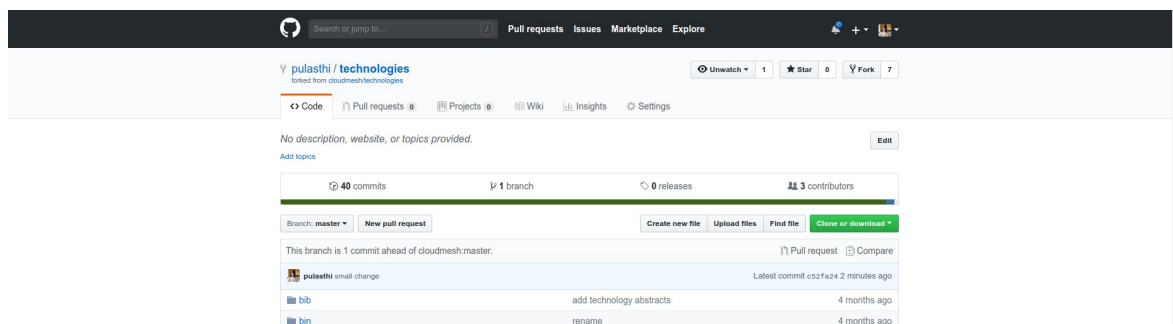


Figure 2: Button Pull request

Once you click on that button you will be taken to a page to create the pull request, which will look similar to [Figure 3](#).

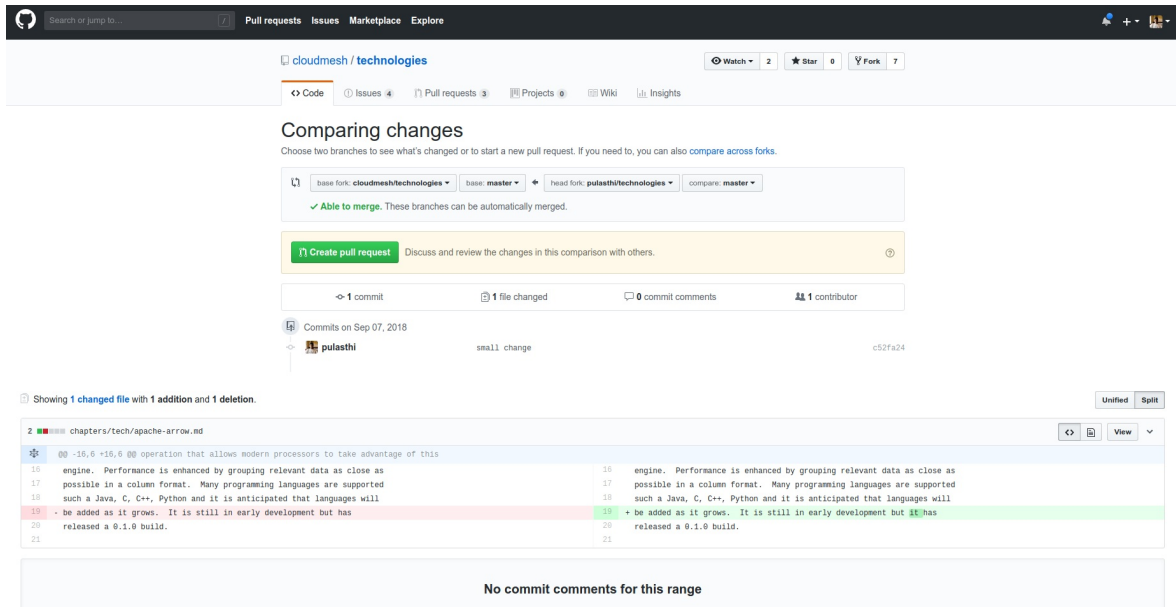


Figure 3: Create a pull request

Once you click on the `create pull request` button you will be given an option to add a title and a comment for the pull request. Once you complete the details and submit the pull request will appear in the original `cloudmesh/technologies` repo.

Note: Make sure you see the `Able to merge` sign before you submit the pull request, otherwise your pull will not be able to directly merged to the original repo. If you do not see this that means you have not properly done the `git pull upstream master` command before you made the changes

 [git example on CL 10:09](#)

3.6.3 Tig

Many browsers exist to gain insight into git repositories. In case you have Linux or Ubuntu a tool to display information in a terminal is available.

- <https://jonas.github.io/tig/>

On OSX it can be installed with:

```
$ brew install tig
```

Tig has many different views including views for main, log, diff, tree, blob,

blame, refs, status, stage. stash, grep, and pager .

A screenshot shows some of its basic functionality is shown in [Figure 4](#)

```

book — tig — 80x24
2019-07-23 11:01 -0500 Unknown o Unstaged changes
2019-07-23 11:01 -0500 Unknown o Staged changes
2019-07-23 12:01 -0400 Gregor von Laszewski o [master] {origin/master} {origi
2019-07-23 12:00 -0400 Gregor von Laszewski o add git tag
2019-07-04 11:13 -0400 Gregor von Laszewski o update makefiles
2019-07-03 15:59 -0400 Gregor von Laszewski o add docs folder
2019-07-03 15:55 -0400 Gregor von Laszewski o add table seperator
2019-07-03 15:50 -0400 Gregor von Laszewski o remove the emojis
2019-07-03 15:24 -0400 Gregor von Laszewski o add syllabus created with cyber
2019-07-03 15:18 -0400 Gregor von Laszewski o add better doc location
2019-07-03 15:13 -0400 Gregor von Laszewski o Create README.yml
2019-07-03 13:21 -0400 Gregor von Laszewski o first updates for fall 2019
2019-06-20 13:37 -0400 Gregor von Laszewski o update epub
2019-06-20 13:37 -0400 Gregor von Laszewski o update epub
2019-05-12 16:09 -0400 Gregor von Laszewski o change python version
2019-04-26 13:27 -0400 Gregor von Laszewski M Merge pull request #457 from
2019-04-26 06:50 -0400 garbeandy | o Update products.md
2019-04-23 01:02 -0400 Gregor von Laszewski M Merge pull request #454 from
2019-04-20 10:06 -0500 Mallik Challa | o Update ai-rest.md
2019-04-22 19:18 -0400 Gregor von Laszewski M Merge pull request #455 fro
2019-04-22 15:39 -0700 Anthony Duer | o Minor formatting update.
2019-04-20 02:26 -0400 Gregor von Laszewski o Update amazon-emr-1.md
[main] Unstaged changes 0%
Cannot move beyond the first line

```

Figure 4: Git tig main vie

Example infocations are

```

$ tig
$ git show | tig
$ git log | tig

```

3.7 LINUX SHELL



Learning Objectives

- Be able to know the basic commands to work in a Linux terminal.
- Get familiar with Linux Commands

In this chapter we introduce you to a number of useful shell commands. You may ask:

“Why is he so keen on telling me all about shells as I do have a beautiful GUI?”

You will soon learn that A GUI may not be that suitable if you like to manage 10, 100, 1000, 10000, ... virtual machines. A commandline interface could be much simpler and would allow scripting.

3.7.1 History

LINUX is a reimplementation by the community of UNIX which was developed in 1969 by Ken Thompson and Dennis Ritchie of Bell Laboratories and rewritten in C. An important part of UNIX is what is called the *kernel* which allows the software to talk to the hardware and utilize it.

In 1991 Linus Torvalds started developing a Linux Kernel that was initially targeted for PC's. This made it possible to run it on Laptops and was later on further developed by making it a full Operating system replacement for UNIX.

3.7.2 Shell

One of the most important features for us will be to access the computer with the help of a *shell*. The shell is typically run in what is called a terminal and allows interaction to the computer with commandline programs.

There are many good tutorials out there that explain why one needs a linux shell and not just a GUI. Randomly we picked the first one that came up with a google query. This is not an endorsement for the material we point to, but could be a worth while read for someone that has no experience in Shell programming:

http://linuxcommand.org/lc3_learning_the_shell.php

Certainly you are welcome to use other resources that may suite you best. We will however summarize in table form a number of useful commands that you may also find even as a RefCard.

<http://www.cheat-sheets.org/#Linux>

We provide in the next table a number of useful commands that you want to explore. For more information simply type man and the name of the command. If you find a useful command that is missing, please add it with a Git pull request.

Command	Description
<code>man <i>command</i></code>	manual page for the <i>command</i>
<code>apropos <i>text</i></code>	list all commands that have <i>text</i> in it
<code>ls</code>	Directory listing
<code>ls -lisa</code>	list details
<code>tree</code>	list the directories in graphical form
<code>cd <i>dirname</i></code>	Change directory to <i>dirname</i>
<code>mkdir <i>dirname</i></code>	create the directory
<code>rmdir <i>dirname</i></code>	delete the directory
<code>pwd</code>	print working directory
<code>rm <i>file</i></code>	remove the file
<code>cp <i>a b</i></code>	copy file <i>a</i> to <i>b</i>
<code>mv <i>a b</i></code>	move/rename file <i>a</i> to <i>b</i>
<code>cat <i>a</i></code>	print content of file <i>a</i>
<code>cat -n <i>filename</i></code>	print content of file <i>a</i> with line numbers
<code>less <i>a</i></code>	print paged content of file <i>a</i>
<code>head -5 <i>a</i></code>	Display first 5 lines of file <i>a</i>
<code>tail -5 <i>a</i></code>	Display last 5 lines of file <i>a</i>
<code>du -hs .</code>	show in human readable form the space used by the current directory
<code>df -h</code>	show the details of the disk file system
<code>wc <i>filename</i></code>	counts the word in a file
<code>sort <i>filename</i></code>	sorts the file
<code>uniq <i>filename</i></code>	displays only uniq entries in the file
<code>tar -xvf <i>dir</i></code>	tars up a compressed version of the directory
<code>rsync</code>	faster, flexible replacement for rcp

<i>gzip filename</i>	compresses the file
<i>gunzip filename</i>	compresses the file
<i>bzip2 filename</i>	compresses the file with block-sorting
<i>bunzip2 filename</i>	uncompresses the file with block- sorting
clear	clears the terminal screen
<i>touch filename</i>	change file access and modification times or if file does not exist creates file
who	displays a list of users that are currently logged on, for each user the login name, date and time of login, tty name, and hostname if not local are displayed
whoami	displays the users effective id see also id
<i>echo -n string</i>	write specified arguments to standard output
date	displays or sets date & time, when invoked without arguments the current date and time are displayed
logout	exit a given session
exit	when issued at the shell prompt the shell will exit and terminate any running jobs within the shell
kill	terminate or signal a process by sending a signal to the specified process usually by the pid
ps	displays a header line followed by all processes that have controlling terminals
sleep	suspends execution for an interval of

	time specified in seconds
uptime	displays how long the system has been running
time <i>command</i>	times the command execution in seconds
find / [-name] <i>file-name.txt</i>	searches a specified path or directory with a given expression that tells the find utility what to find, if used as shown the find utility would search the entire drive for a file named file-name.txt
diff	compares files line by line
hostname	prints the name of the current host system
which	locates a program file in the users path
tail	displays the last part of the file
head	displays the first lines of a file
top	displays a sorted list of system processes
locate <i>filename</i>	finds the path of a file
grep 'word' <i>filename</i>	finds all lines with the word in it
grep -v 'word' <i>filename</i>	finds all lines without the word in it
chmod ug+rw <i>filename</i>	change file modes or Access Control Lists. In this example user and group are changed to read and write
chown	change file owner and group
history	a build-in command to list the past commands
sudo	execute a command as another user
su	substitute user identity
uname	print the operating system name

set -o emacs	tells the shell to use Emacs commands.
chmod go-rwx <i>file</i>	changes the permission of the file
chown <i>username file</i>	changes the ownership of the file
chgrp <i>group file</i>	changes the group of a file
fgrep <i>text filename</i>	searches the text in the given file
grep -R <i>text .</i>	recursively searches for xyz in all files
find . -name *.py	find all files with .py at the end
ps	list the running processes
kill -9 1234	kill the process with the id 1234
at	que commands for later execution
cron	daemon to execute scheduled commands
crontab	manage the time table for execution commands with cron
mount /dev/cdrom /mnt/cdrom	mount a filesystem from a cd rom to /mnt/cdrom
users	list the logged in users
who	display who is logged in
whoami	print the user id
dmesg	display the system message buffer
last	indicate last logins of users and ttys
uname	print operating system name
date	prints the current date and time
time <i>command</i>	prints the sys, real and user time
shutdown -h "shut down"	shutdown the computer
ping	ping a host
netstat	show network status
hostname	print name of current host system
	print the route packets take to

traceroute	network host
ifconfig	configure network interface parameters
host	DNS lookup utility
whois	Internet domain name and network number directory service
dig	DNS lookup utility
wget	non-interactive network downloader
curl	transfer a URL
ssh	remote login program
scp	remote file copy program
sftp	secure file transfer program
watch <i>command</i>	run any designated command at regular intervals
awk	program that you can use to select particular records in a file and perform operations on them
sed	stream editor used to perform basic text transformations
xargs	program that can be used to build and execute commands from STDIN
cat <i>some_file.json</i> python -m json.tool	quick and easy JSON validator

3.7.3 The command man

On Linux you find a rich set of manual pages for these commands. Try to pick one and execute:

```
$ man ls
```

You will see something like this

```
LS(1)
```

```
BSD General Commands Manual
```

```
LS(1)
```

```
NAME
```

```
ls -- list directory contents
```

SYNOPSIS

```
ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1] [file ...]
```

DESCRIPTION

For each operand that names a file of a type other than directory, `ls` displays its name as well as any requested, associated information. For each operand that names a file of type directory, `ls` displays the names of files contained within that directory, as well as any requested, associated information.

If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

The following options are available:

- @ Display extended attribute keys and sizes in long (-l) output.
- 1 (The numeric digit 'one'.) Force output to be one entry per line. This is the default when output is not to a terminal.
- A List all entries except for . and ... Always set for the super-user.
- a Include directory entries whose names begin with a dot (.).

... on purpose cut ... instead try it yourslef

3.7.4 Multi-command execution

One of the important features is that one can execute multiple commands in the shell.

To execute command 2 once command 1 has finished use

```
command1; command2
```

To execute command 2 as soon as command 1 forwards output to stdout use

```
command1; command2
```

To execute command 1 in the background use

```
command1 &
```

3.7.5 Keyboard Shortcuts

These shortcuts will come in handy. Note that many overlap with emacs shortcuts.

.

Keys	Description
Up Arrow	Show the previous command
Ctrl + z	Stops the current command
	Resume with fg in the foreground
	Resume with bg in the background
Ctrl + c	Halts the current command
Ctrl + l	Clear the screen
Ctrl + a	Return to the start of the line
Ctrl + e	Go to the end of the line
Ctrl + k	Cut everything after the cursor to a special clipboard
Ctrl + y	Paste from the special clipboard
Ctrl + d	Logout of current session, similar to exit

3.7.6 bashrc, bash_profile or zprofile

Usage of a particular command and all the attributes associated with it, use `man` command. Avoid using `rm -r` command to delete files recursively. A good way to avoid accidental deletion is to include the following in the file `.bash_profile` or `.zprofile` on macOS or `.bashrc` on other platforms:

```
alias rm='rm -i'  
alias mv='mv -i'  
alias h='history'
```

3.7.7 Makefile

Makefiles allow developers to coordinate the execution of code compilations. This not only includes C or C++ code, but any translation from source to a final format. For us this could include the creation of PDF files from latex sources, creation of docker images, and the creation of cloud services and their deployment through simple workflows represented in makefiles, or the coordination of execution targets.

As makefiles include a simple syntax allowing structural dependencies they can easily adapted to fulfill simple activities to be executed in repeated fashion by developers.

An example of how to use Makefiles for docker is provided at <http://jmkhael.io/makefiles-for-your-dockerfiles/>.

An example on how to use Makefiles for LaTeX is provided at <https://github.com/cloudmesh/book/blob/master/Makefile>.

Makefiles include a number of rules that are defined by a target name. Let us define a target called hello that prints out the string “Hello World”.

```
hello:
    @echo "Hello World"
```

Important to remember is that the commands after a target are not indented just by spaces, but actually by a single TAB character. Editors such as emacs will be ideal to edit such Makefiles, while allowing syntax highlighting and easy manipulation of TABs. Naturally other editors will do that also. Please chose your editor of choice. One of the best features of targets is that they can depend on other targets. Thus, iw we define

```
hallo: hello
    @echo "Hallo World"
```

our makefile will first execute hello and than all commands in hallo. As you can see this can be very useful for defining simple dependencies.

In addition we can define variables in a makefile such as

```
HELLO="Hello World"

hello:
    @echo $(HELLO)
```

and can use them in our text with \$ invocations.

Moreover, in sophisticated Makefiles, we could even make the targets dependent on files and a target rules could be defined that only compiles those files that have changed since our last invocation of the Makefile, saving potentially a lot of time. However, for our work here we just use the most elementary makefiles.

For more information we recommend you to find out about it on the internet. A convenient reference card is available at <http://www.cs.jhu.edu/~joanne/unixRC.pdf>.

3.7.8 chmod

The `chmod` command stand for *change mode* and changes the access permissions for a given file system object(s). It uses the following syntax: `chmod [options] mode[,mode] file1 [file2...]`. The option parameters modify how the process runs, including what information is outputted to the shell:

Option:	Description:
<code>-f, --silent, --quiet</code>	Forces process to continue even if errors occur
<code>-v, --verbose</code>	Outputs for every file that is processed
<code>-c, --changes</code>	Outputs when a file is changed
<code>--reference=RFile</code>	Uses RFile instead of Mode values
<code>-R, --recursive</code>	Make changes to objects in subdirectories as well
<code>--help</code>	Show help
<code>--version</code>	Show version information

Modes specify which rights to give to which users. Potential users include the user who owns the file, users in the file's Group, other users not in the file's Group, and all, and are abbreviated as `u`, `g`, `o`, and `a` respectively. More than one user can be specified in the same command, such as `chmod -v ug(operator)(permissions) file.txt`. If no user is specified, the command defaults to `a`. Next, a `+` or `-` indicates whether permissions should be added or removed for the selected user(s). The permissions are as follows:

Permission:	Description:
<code>r</code>	Read
<code>w</code>	Write
<code>x</code>	Execute file or access directory
<code>X</code>	Execute only if the object is a directory
<code>s</code>	Set the user or group ID when running
<code>t</code>	Restricted deletion flag or sticky mode
<code>u</code>	Specifies the permissions the user who owns the file has
<code>g</code>	

Specifies the permissions of the group

o

Specifies the permissions of users not in the group

More than one permission can be also be used in the same command as follows:

```
$ chmod -v o+rw file.txt
```

Multiple files can also be specified:

```
$ chmod a-x,o+r file1.txt file2.txt
```

3.7.9 Exercises

E.Linux.1

Familiarize yourself with the commands

E.Linux.2

Find more commands that you find useful and add them to this page.

E.Linux.3

Use the sort command to sort all lines of a file while removing duplicates.

E.Linux.4

Should there be other commands listed in the table with the Linux commands If so which? Create a pull request for them.

E.Linux.5

Write a section explaining chmod. Use letters not numbers

E.Linux.6

Write a section explaining chown. Use letters not numbers

E.Linux.7

Write a section explaining su and sudo

E.Linux.8

Write a section explaining cron, at, and crontab

3.8 SECURE SHELL



Learning Objectives

- This is a very important sections of the book, studdy it carefully.
 - learn how to use SSH keys
 - Learn how to use ssh-add and ssh-keycahin so you only have to type in your password once
 - Understand that each computer needs its own ssh key
-

[Secure Shell](#) is a network protocol allowing users to securely connect to remote resources over the internet. In many services we need to use SSH to assure that we protect he messages send between the communicating entities. Secure Shell is based on public key technology requiring to generate a public-private key pair on the computer. The public key will than be uploaded to the remote machine and when a connection is established during authentication the public private key pair is tested. If they match authentication is granted. As many users may have to share a computer it is possible to add a list of public keys so that a number of computers can connect to a server that hosts such a list. This mechanism builds the basis for networked computers.

In this section we will introduce you to some of the commands to utilize secure shell. We will reuse this technology in other sections to for example create a network of workstations to which we can log in from your laptop. For more information please also consult with the [SSH Manual](#).



Whatever others tell you, the private key should never be copied to another machine. You almost always want to have a passphrase

protecting your key.

3.8.1 ssh-keygen

The first thing you will need to do is to create a public private key pair. Before you do this check whether there are already keys on the computer you are using:

```
ls ~/.ssh
```

If there are files named `id_rsa.pub` or `id_dsa.pub`, then the keys are set up already, and we can skip the generating keys step. However you must know the passphrase of the key. If you forgot it you will need to recreate the key. However you will lose any ability to connect with the old key to the resources to which you uploaded the public key. So be careful.

To generate a key pair use the command [ssh-keygen](#). This program is commonly available on most UNIX systems and most recently even Windows 10.

To generate the key, please type:

```
$ ssh-keygen -t rsa -C <comment>
```

The comment will remind you where the key has been created, you could for example use the hostname on which you created the key.

In the following text we will use *localname* to indicate the username on your computer on which you execute the command.

The command requires the interaction of the user. The first question is:

```
Enter file in which to save the key (/home/localname/.ssh/id_rsa):
```

We recommend using the default location `~/.ssh/` and the default name `id_rsa`. To do so, just press the enter key.

The second and third question is to protect your ssh key with a passphrase. This passphrase will protect your key because you need to type it when you want to use it. Thus, you can either type a passphrase or press enter to leave it without passphrase. To avoid security problems, you **MUST** chose a passphrase.

It will ask you for the location and name of the new key. It will also ask you for a passphrase, which you **MUST** provide. Please use a strong passphrase to protect it appropriately. Some may advise you (including teachers and TA's) to not use passphrases. This is **WRONG** as it allows someone that gains access to your computer to also gain access to all resources that have the public key. Only for some system related services you may create passwordless keys, but such systems need to be properly protected.



Not using passphrases poses a security risk!

Make sure to not just type return for an empty passphrase:

```
Enter passphrase (empty for no passphrase):
```

and:

```
Enter same passphrase again:
```

If executed correctly, you will see some output similar to:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/localname/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/localname/.ssh/id_rsa.
Your public key has been saved in /home/localname/.ssh/id_rsa.pub.
The key fingerprint is:
34:87:67:ea:c2:49:ee:c2:81:d2:10:84:b1:3e:05:59 localname@indiana.edu
```

```
+--[ RSA 2048]-----+
|..Eo= . |
|..= .o + o +o |
|0. = ..... |
| = . . |
+-----+

```


Once, you have generated your key, you should have them in the `.ssh` directory. You can check it by:

```
$ cat ~/.ssh/id_rsa.pub
```

If everything is normal, you will see something like:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACXJH2iG2FMHqC6T/U7uB8kt
6K1Rh4kU0jgw9sc4Uu+Uwe/kshuispauhfshfm,anf6787sjgdkjsgl+EwD0
thkoamyi0VvhTVzhj61pTdhyl1t8h1koL19JvnVBPP5kIN3wVyNAJjYBrAUNW
4dXKXtmfKxp98T30W4mxAtTH434MaT+QcPTcxims/hwsUeDAVKZY7UgZhEbiE
xxkejtnRBHTipi0W03W05TOUGRW7EuKf/4ftNVPi1C04DpfY44NFG1xPwHeim
Uk+t9h48pBQj16FrUCp0rS02Pj+4/9dNeS1kmNJUSZYS8HVRhvu0TXuAY/UVc
ynEPUegkp+qYnR user@myemail.edu
```

The directory `~/.ssh` will also contain the private key `id_rsa` which you must not share or copy to another computer.

 *Never, copy your private key to another machine or check it into a repository!*

To see what is in the `.ssh` directory, please use

```
$ ls ~/.ssh
```

Typically you will see a list of files such as

```
authorized_keys
id_rsa
id_rsa.pub
known_hosts
```

In case you need to change your change passphrase, you can simply run `ssh-keygen -p` command. Then specify the location of your current key, and input (old and) new passphrases. There is no need to re-generate keys:

```
ssh-keygen -p
```

You will see the following output once you have completed that step:

```
Enter file in which the key is (/home/localname/.ssh/id_rsa):
Enter old passphrase:
Key has comment '/home/localname/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

3.8.2 ssh-add

Often you will find wrong information about passphrases on the internet and people recommending you not to use one. However it is in almost all cases better to create a key pair and use `ssh-add` to add the key to the current session so it can be used in behalf of you. This is accomplished with an agent.

The `ssh-add` command adds SSH private keys into the SSH authentication agent for implementing single sign-on with SSH. `ssh-add` allows the user to use any number of servers that are spread across any number of organizations, without having to type in a password every time when connecting between servers. This

is commonly used by system administrators to login to multiple server.

`ssh-add` can be run without arguments. When run without arguments, it adds the following default files if they do exist:

- `~/.ssh/identity` - Contains the protocol version 1 RSA authentication identity of the user.
- `~/.ssh/id_rsa` - Contains the protocol version 1 RSA authentication identity of the user.
- `~/.ssh/id_dsa` - Contains the protocol version 2 DSA authentication identity of the user.
- `~/.ssh/id_ecdsa` - Contains the protocol version 2 ECDSA authentication identity of the user.

To add a key you can provide the path of the key file as an argument to `ssh-add`. For example,

```
ssh-add ~/.ssh/id_rsa
```

would add the file `~/.ssh/id_rsa`

If the key being added has a passphrase, `ssh-add` will run the `ssh-askpass` program to obtain the passphrase from the user. If the `SSH_ASKPASS` environment variable is set, the program given by that environment variable is used instead.

Some people use the `SSH_ASKPASS` environment variable in scripts to provide a passphrase for a key. The passphrase might then be hard-coded into the script, or the script might fetch it from a password vault.

The command line options of `ssh-add` are as follows:

Option	Description
<code>-c</code>	Causes a confirmation to be requested from the user every time the added identities are used for authentication. The confirmation is requested using <code>ssh-askpass</code> .
<code>-D</code>	Deletes all identities from the agent.
<code>-d</code>	Deletes the given identities from the agent. The private key files for the identities to be deleted should be listed on the

command line.

<code>-e pkcs11</code>	Remove key provided by pkcs11
<code>-L</code>	Lists public key parameters of all identities currently represented by the agent.
<code>-l</code>	Lists fingerprints of all identities currently represented by the agent.
<code>-s pkcs11</code>	Add key provided by pkcs11.
<code>-t life</code>	Sets the maximum time the agent will keep the given key. After the timeout expires, the key will be automatically removed from the agent. The default value is in seconds, but can be suffixed for m for minutes, h for hours, d for days, or w for weeks.
<code>-X</code>	Unlocks the agent. This asks for a password to unlock.
<code>-x</code>	Locks the agent. This asks for a password; the password is required for unlocking the agent. When the agent is locked, it cannot be used for authentication.

3.8.3 SSH Add and Agent

To not always type in your password, you can use `ssh-add` as previously discussed

It prompts the user for a private key passphrase and add it to a list of keys managed by the ssh-agent. Once it is in this list, you will not be asked for the passphrase as long as the agent is running with your public key. To use the key across terminal shells you can start an ssh agent.

To start the agent please use the following command:

```
$ eval `ssh-agent`
```

or use

```
$ eval "$(ssh-agent -s)"
```

It is important that you use the backquote, located under the tilde (US keyboard), rather than the single quote. Once the agent is started it will print a PID that you can use to interact with later

To add the key use the command

```
$ ssh-add
```

To remove the agent use the command

```
kill $SSH_AGENT_PID
```

To execute the command upon logout, place it in your `.bash_logout` (assuming you use bash).

On OSX you can also add the key permanently to the keychain if you do the following:

```
ssh-add -K ~/.ssh/id_rsa
```

Modify the file `~/.ssh/config` and add the following lines:

```
Host *
  UseKeychain yes
  AddKeysToAgent yes
  IdentityFile ~/.ssh/id_rsa
```

3.8.3.1 Using SSH on Mac OS X

Mac OS X comes with an ssh client. In order to use it you need to open the `Terminal.app` application. Go to `Finder`, then click `Go` in the menu bar at the top of the screen. Now click `utilities` and then open the `Terminal` application.

3.8.3.2 Using SSH on Linux

All Linux versions come with ssh and can be used right from the terminal.

3.8.3.3 Using SSH on Raspberry Pi 3/4

SSH is available on Raspbian. However, to ssh into the PI you have to activate it via the configuration menu.

3.8.3.4 Accessing a Remote Machine

Once the key pair is generated, you can use it to access a remote machine. To

do so the public key needs to be added to the `authorized_keys` file on the remote machine.

The easiest way to do this is to use the command `ssh-copy-id`.

```
$ ssh-copy-id user@host
```

Note that the first time you will have to authenticate with your password.

Alternatively, if the `ssh-copy-id` is not available on your system, you can copy the file manually over SSH:

```
$ cat ~/.ssh/id_rsa.pub | ssh user@host 'cat >> ~/.ssh/authorized_keys'
```


Now try:

```
$ ssh user@host
```

and you will not be prompted for a password. However, if you set a passphrase when creating your SSH key, you will be asked to enter the passphrase at that time (and whenever else you log in in the future). To avoid typing in the password all the time we use the `ssh-add` command that we described earlier.

```
$ ssh-add
```

3.8.4 SSH Port Forwarding

 **TODO:** Add images to illustrate the concepts

SSH Port forwarding (SSH tunneling) creates an encrypted secure connection between a local computer and a remote computer through which services can be relayed. Because the connection is encrypted, SSH tunneling is useful for transmitting information that uses an unencrypted protocol.

3.8.4.1 Prerequisites

- Before you begin, you need to check if forwarding is allowed on the SSH server you will connect to.
- You also need to have a SSH client on the computer you are working on.

If you are using the OpenSSH server:

```
$ vi /etc/ssh/sshd_config
```

and look and change the following:

```
AllowTcpForwarding = Yes  
GatewayPorts = Yes
```

Set the `GatewayPorts` variable only if you are going to use remote port forwarding (discussed later in this tutorial). Then, you need to restart the server for the change to take effect.

3.8.4.2 How to Restart the Server

If you are on:

- Linux, depending upon the init system used by your distribution, run:

```
$ sudo systemctl restart sshd  
$ sudo service sshd restart
```

Note that depending on your distribution, you may have to change the service to `ssh` instead of `sshd`.

- Mac, you can restart the server using:

```
$ sudo launchctl unload /System/Library/LaunchDaemons/ssh.plist  
$ sudo launchctl load -w /System/Library/LaunchDaemons/ssh.plist
```

- Windows and want to set up a SSH server, have a look at MSYS2 or Cygwin.

3.8.4.3 Types of Port Forwarding

There are three types of SSH Port forwarding:

3.8.4.4 Local Port Forwarding

Local port forwarding lets you connect from your local computer to another server. It allows you to forward traffic on a port of your local computer to the SSH server, which is forwarded to a destination server. To use local port forwarding, you need to know your destination server, and two port numbers.

Example 1:

```
$ ssh -L 8080:www.cloudcomputing.org:80 <host>
```

Where `<host>` should be replaced by the name of your laptop. The `-L` option specifies local port forwarding. For the duration of the SSH session, pointing your browser at `http://localhost:8080/` would send you to `http://cloudcomputing.com`

Example 2:

This example opens a connection to the `www.cloudcomputing.com` jump server, and forwards any connection to port 80 on the local machine to port 80 on `intra.example.com`.

```
$ ssh -L 80:intra.example.com:80 www.cloudcomputing.com
```

Example 3:

By default, anyone (even on different machines) can connect to the specified port on the SSH client machine. However, this can be restricted to programs on the same host by supplying a bind address:

```
$ ssh -L 127.0.0.1:80:intra.example.com:80 www.cloudcomputing.com
```

Example 4:

```
$ ssh -L 8080:www.Cloudcomputing.com:80 -L 12345:cloud.com:80 <host>
```

This would forward two connections, one to `www.cloudcomputing.com`, the other to `www.cloud.com`. Pointing your browser at `http://localhost:8080/` would download pages from `www.cloudcomputing.com`, and pointing your browser to `http://localhost:12345/` would download pages from `www.cloud.com`.

Example 5:

The destination server can even be the same as the SSH server.

```
$ ssh -L 5900:localhost:5900 <host>
```

The `LocalForward` option in the OpenSSH client configuration file can be used to configure forwarding without having to specify it on command line.

3.8.4.5 Remote Port Forwarding

Remote port forwarding is the exact opposite of local port forwarding. It forwards traffic coming to a port on your server to your local computer, and then it is sent to a destination. The first argument should be the remote port where traffic will be directed on the remote system. The second argument should be the address and port to point the traffic to when it arrives on the local system.

```
$ ssh -R 9000:localhost:3000 user@cloudcomputing.com
```

SSH does not by default allow remote hosts to forwarded ports. To enable remote forwarding add the following to: `/etc/ssh/sshd_config`

```
GatewayPorts yes
```

```
$ sudo vim /etc/ssh/sshd_config
```

and restart SSH

```
$ sudo service ssh restart
```

After completing the previous steps you should be able to connect to the server remotely, even from your local machine. `ssh -R` first creates an SSH tunnel that forwards traffic from the server on port 9000 to your local machine on port 3000.

3.8.4.6 Dynamic Port Forwarding

Dynamic port forwarding turns your SSH client into a SOCKS proxy server. SOCKS is a little-known but widely-implemented protocol for programs to request any Internet connection through a proxy server. Each program that uses the proxy server needs to be configured specifically, and reconfigured when you stop using the proxy server.

```
$ ssh -D 5000 user@cloudcomputing.com
```

The SSH client creates a SOCKS proxy at port 5000 on your local computer. Any traffic sent to this port is sent to its destination through the SSH server.

Next, you'll need to configure your applications to use this server. The *Settings* section of most web browsers allow you to use a SOCKS proxy.

3.8.4.7 ssh config

Defaults and other configurations can be added to a configuration file that is placed in the system. The ssh program on a host receives its configuration from

- the command line options
- a user-specific configuration file: `~/.ssh/config`
- a system-wide configuration file: `/etc/ssh/ssh_config`

Next we provide an example on how to use a config file

3.8.4.8 Tips

Use SSH keys

- You will need to use ssh keys to access remote machines

No blank passphrases

- In most cases you must use a passphrase with your key. In fact if we find that you use passwordless keys to futuresystems and to chameleon cloud resources, we may elect to give you an *F* for the assignment in question. There are some exceptions, but they will be clearly communicated to you in class. You will as part of your cloud drivers license test explain how you gain access to futuresystems and chameleon to explicitly explain this point and provide us with reasons what you can not do.

A key for each server

- Under no circumstances copy the same private key on multiple servers. This violates security best practices. Create for each server a new private key and use their public keys to gain access to the appropriate server.

Use SSH agent

- So as to not to type in all the time the passphrase for a key, we recommend using ssh-agent to manage the login. This will be part of your cloud drivers license.

But shut down the ssh-agent if not in use

keep an offline backup, put encrypt the drive

- You may for some of our projects need to make backups of private keys on other servers you set up. If you like to make a backup you can do so on a USB stick, but make sure that access to the stick is encrypted. Do not store anything else on that key and look it in a safe place. If you lose the stick, recreate all keys on all machines.

3.8.4.9 References

- [The Secure Shell: The Definitive Guide, 2 Ed \(O'Reilly and Associates\)](#)

3.8.5 SSH to FutureSystems Resources



Learning Objectives

- Obtain a Future system account so you can use kubernetes or dockerswarm or other services offered by FutureSystems.
-

Next, you need to upload the key to the portal. You must be logged into the portal to do so.

Step 1: Log into the portal <https://portal.futuresystems.org/>

Step 2: Click on the “MY ACCOUNT” link.

Step 3: Click on “EDIT”

Step 4: Paste your ssh key into the box marked Public SSH Key. Use a text editor to open the `id_rsa.pub`. Copy the entire contents of this file into the ssh key field as part of your profile information. Many errors are introduced by users in this step as they do not paste and copy correctly.

If you need to add keys, use the Add another item button

At this point, you have uploaded your key. However, you will still need to wait till all accounts have been set up to use the key, or if you did not have an account till it has been created by an administrator. Please, check your email for further updates. You can also refresh this page and see if the boxes in your account status information are all green. Then you can continue.

3.8.5.1 Testing your FutureSystems ssh key

If you have had no FutureSystem account before, you need to wait for up to two business days so we can verify your identity and create the account. So please wait. Otherwise, testing your new key is almost instantaneous on india. For other clusters like it can take around 30 minutes to update the ssh keys.

To log into india simply type the usual ssh command such as:

```
$ ssh portalname@india.futuresystems.org
```

The first time you ssh into a machine you will see a message like this:

```
The authenticity of host 'india.futuresystems.org (192.165.148.5)' cannot be established.  
RSA key fingerprint is 11:96:de:b7:21:eb:64:92:ab:de:e0:79:f3:fb:86:dd.  
Are you sure you want to continue connecting (yes/no)? yes
```

You have to type yes and press enter. Then you will be logging into india. Other FutureSystem machines can be reached in the same fashion. Just replace the name india, with the appropriate FutureSystems resource name.

3.8.6 Exercises

E.SSH.1:

Create an SSH key pair

E.SSH.2:

Upload the public key to git repository you use.

E.SSH.3:

What is the output of a key that has a passphrase when executing the following command. Test it out on your key

```
$ grep ENCRYPTED ~/.ssh/id_rsa
```

E.SSH.4

Get an account on futuresystems.org (if you are authorized to do so). Upload your key to <https://futuresystems.org>. Login to india.futuresystems.org. Note that this could take some time as administrators need to approve you. Be patient.

E.SSH.5:

What can happen if you copy your private key to a machine on the network?

E.SSH.6:

Should I share my private key with others?

E.SSH.7:

Assume I participate in a video conference call and I accidentally share my private key. What should I do?

E.SSH.8:

Assume I participate in a video conference call and I accidentally share my public key. What should I do?

4 DEVOPS

4.1 DEVOPS

When we work with cloud infrastructure and at the same time deploy platforms and software we become in essence our own system administrator. Today the intervoven responsibilities of the developer not only to design an algorithm but be familiar with how this algorithm can be effectively deployed on an infrastructure is a challange that Big Data Scientists, Big Data Engineers and Cloud researchers must master.

A field of study that will be helpful in achieving this is called *DevOps*.

We quote form <https://en.wikipedia.org/wiki/DevOps>:

DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.[8]

[8] Bass, Len; Weber, Ingo; Zhu, Liming. *DevOps: A Software Architect's Perspective*. ISBN 978-0134049847.

A helpful ven diagram for visualizing the important aspects that DevOps tries to address is depicted in [Figure 5](#)

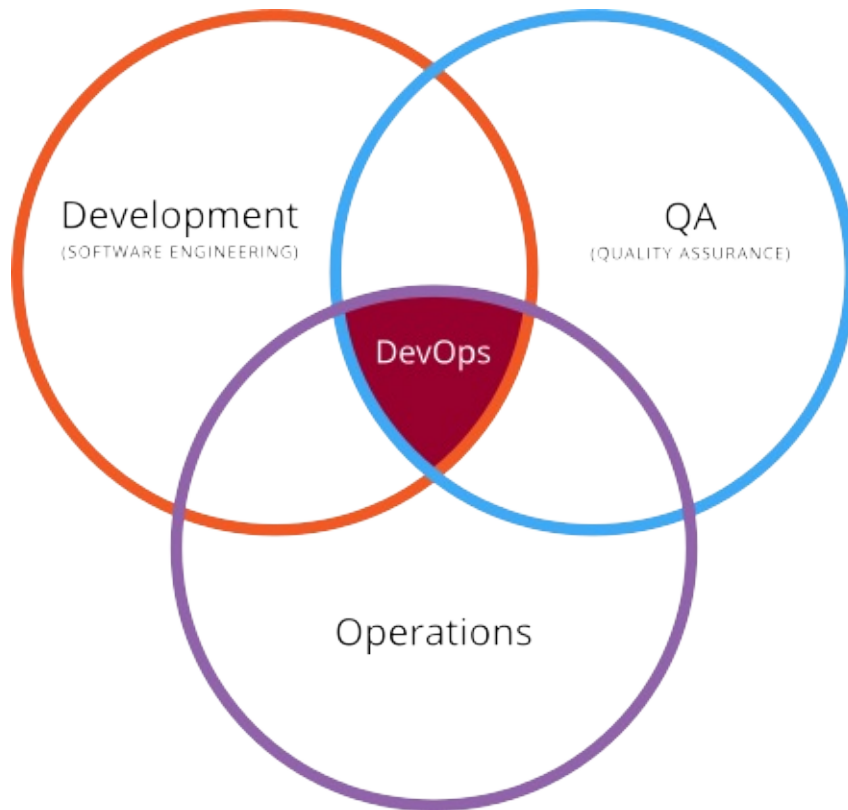


Figure 5: Wikipedia Ven Diagram about DevOps

Source: <https://en.wikipedia.org/wiki/DevOps#/media/File:Devops.svg>

To visualize the process interactions in the toolchain cycle between development and operations we also often find it depicted in [Figure 6](#)

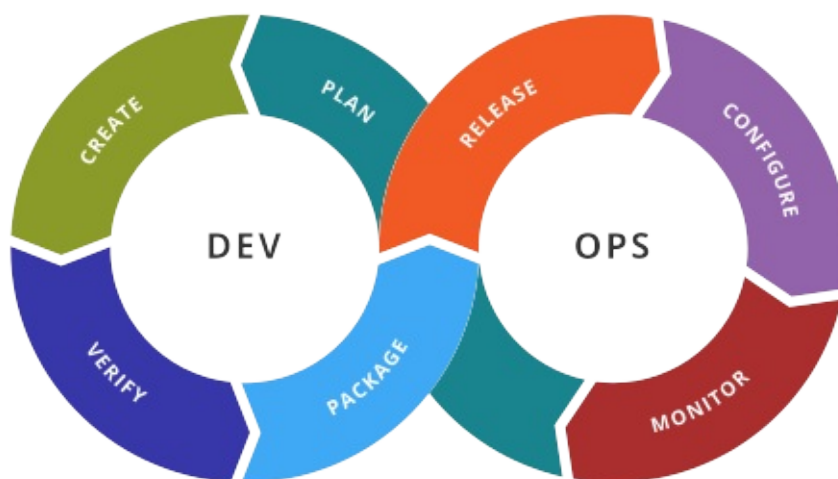


Figure 6: Wikipedia Ven Diagram about DevOps

The cycle includes in each portion

Dev:

- dev planing
- creating
- verifying
- packaging

Ops:

- releasing
- configuring
- monitoring
- ops planing (influencing dev planing)

One of the advantages we have while adopting a DevOps approach is to also address reproducibility across the entire product chain. That is if we apply these technologies correctly we should be able to adapt our DevOps solutions in such a form that it not only works on one cloud when we develop general tools, but also on multiple clouds. We can enable this through different entry points into the toolchain while for example using a common underlying infrastructure, platform, or software environment to address each DevOps challenge in that particular layer.

4.1.1 References

- DevOps tools <https://github.com/collections/devops-tools>
- Tox <https://tox.readthedocs.io/en/latest/>
- Travis:
 - <https://about.travis-ci.com/> <https://travis-ci.org/>
 - <https://docs.travis-ci.com/user/getting-started/>

4.2 TRAVIS

Travis CI is a continuous integration tool that is often used as part of DevOps development. It is a hosted service that enables users to test their projects on GitHub.

Once travis is activated in a GitHub project, the developers can place a `.travis` file in the project root. Upon checkin the travis configuration file will be interpreted and the commands indicated in it will be executed.

In fact this book has also a travis file that is located at

- <https://github.com/cloudmesh-community/book/blob/master/.travis.yml>

Please inspect it as we will illustrate some concepts of it. Unfortunately travis does not use an up to date operating system such as ubuntu 18.04. Therefore it contains outdated libraries. Although we would be able to use containers, we have elected for us to chose mechanism to update the operating system as we need.

This is done in the `install` phase that in our case installs a new version of pandoc, as well as some additional libraries that we use.

in the `env` we specify where we can find our executables with the `PATH` variable.

The last portion in our example file specifies the script that is executed after the install phase has been completed. As our installation contains convenient and sophisticated makefiles, the script is very simple while executing the appropriate make command in the corresponding directories.

4.2.1 Exercises

E.travis.1:

Develop an alternative travis file that in conjunction uses a preconfigured container for ubuntu 18.04

E.travis.2:

Develop an travis file that checks our books on multiple operating systems such as macOS, and ubuntu 18.04.

4.2.2 Resources

- <https://docs.travis-ci.com/>

4.3 ANSIBLE

4.3.1 Introduction to Ansible

Ansible is an open-source IT automation DevOps engine allowing you to manage and configure many compute resources in a scalable, consistent and reliable way.

Ansible to automates the following tasks:

- **Provisioning:** It sets up the servers that you will use as part of your infrastructure.
- **Configuration management:** You can change the configuration of an application, OS, or device. You can implement security policies and other configuration tasks.
- **Service management:** You can start and stop services, install updates
- **Application deployment:** You can conduct application deployments in an automated fashion that integrate with your DevOps strategies.

4.3.1.1 Prerequisite

We assume you

- can install Ubuntu 18.04 virtual machine on VirtualBox
- can install software packages via ‘apt-get’ tool in Ubuntu virtual host
- already reserved a virtual cluster (with at least 1 virtual machine in it) on some cloud. OR you can use VMs installed in VirtualBox instead.
- have SSH credentials and can login to your virtual machines.

4.3.1.2 Setting up a playbook

Let us develop a sample from scratch, based on the paradigms that ansible supports. We are going to use Ansible to install Apache server on our virtual machines.

First, we install ansible on our machine and make sure we have an up to date OS:

```
$ sudo apt-get update
$ sudo apt-get install ansible
```

Next, we prepare a working environment for your Ansible example

```
$ mkdir ansible-apache
$ cd ansible-apache
```

To use ansible we will need a local configuration. When you execute Ansible within this folder, this local configuration file is always going to overwrite a system level Ansible configuration. It is in general beneficial to keep custom configurations locally unless you absolutely believe it should be applied system wide. Create a file `inventory.cfg` in this folder, add the following:

```
[defaults]
hostfile = hosts.txt
```

This local configuration file tells that the target machines' names are given in a file named `hosts.txt`. Next we will specify hosts in the file.

You should have ssh login accesses to all VMs listed in this file as part of our prerequisites. Now create and edit file `hosts.txt` with the following content:

```
[apache]
<server_ip> ansible_ssh_user=<server_username>
```

The name `apache` in the brackets defines a server group name. We will use this name to refer to all server items in this group. As we intend to install and run apache on the server, the name choice seems quite appropriate. Fill in the IP addresses of the virtual machines you launched in your VirtualBox and fire up these VMs in you VirtualBox.

To deploy the service, we need to create a playbook. A playbook tells Ansible what to do. it uses YAML Markup syntax. Create and edit a file with a proper name e.g. `apache.yml` as follow:

```
---
```

```
- hosts: apache #comment: apache is the group name we just defined
  become: yes #comment: this operation needs privilege access
  tasks:
    - name: install apache2 # text description
      apt: name=apache2 update_cache=yes state=latest
```

This block defines the target VMs and operations(tasks) need to apply. We are using the `apt` attribute to indicate all software packages that need to be installed. Dependent on the distribution of the operating system it will find the correct module installer without your knowledge. Thus an ansible playbook could also work for multiple different OSes.

Ansible relies on various kinds of modules to fulfil tasks on the remote servers. These modules are developed for particular tasks and take in related arguments. For instance, when we use `apt` module, we need to tell which package we intend to install. That is why we provide a value for the `name=` argument. The first `-name` attribute is just a comment that will be printed when this task is executed.

4.3.1.3 Run the playbook

In the same folder, execute

```
ansible-playbook apache.yml --ask-sudo-pass
```

After a successful run, open a browser and fill in your server IP. you should see an ‘It works!’ Apache2 Ubuntu default page. Make sure the security policy on your cloud opens port 80 to let the HTTP traffic go through.

Ansible playbook can have more complex and fancy structure and syntaxes. Go explore! This example is based on:

- <https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-18-04>

We are going to offer an advanced Ansible in next chapter.

4.3.2 Ansible Roles

Next we install the R package onto our cloud VMs. R is a useful statistic programming language commonly used in many scientific and statistics computing projects, maybe also the one you chose for this class. With this example we

illustrate the concept of Ansible Roles, install source code through Github, and make use of variables. These are key features you will find useful in your project deployments.

We are going to use a top-down fashion in this example. We first start from a playbook that is already good to go. You can execute this playbook (do not do it yet, always read the entire section first) to get R installed in your remote hosts. We then further complicate this concise playbook by introducing functionalities to do the same tasks but in different ways. Although these different ways are not necessary they help you grasp the power of Ansible and ease your life when they are needed in your real projects.

Let us now create the following playbook with the name `example.yml`:

```
---
- hosts: R_hosts
  become: yes
  tasks:
    - name: install the R package
      apt: name=r-base update_cache=yes state=latest
```

The hosts are defined in a file `hosts.txt`, which we configured in a file that we now call `ansible.cfg`:

```
[R_hosts]
<cloud_server_ip> ansible_ssh_user=<cloud_server_username>
```

Certainly, this should get the installation job done. But we are going to extend it via new features called role next

Role is an important concept used often in large Ansible projects. You divide a series of tasks into different groups. Each group corresponds to certain role within the project.

For example, if your project is to deploy a web site, you may need to install the back end database, the web server that responses HTTP requests and the web application itself. They are three different roles and should carry out their own installation and configuration tasks.

Even though we only need to install the R package in this example, we can still do it by defining a role 'r'. Let us modify our `example.yml` to be:

```
---
- hosts: R_hosts
```

```
roles:
  - r
```

Now we create a directory structure in your top project directory as follows

```
$ mkdir -p roles/r/tasks
$ touch roles/r/tasks/main.yml
```

Next, we edit the `main.yml` file and include the following content:

```
---
- name: install the R package
  apt: name=r-base update_cache=yes state=latest
  become: yes
```

You probably already get the point. We take the ‘tasks’ section out of the earlier `example.yml` and re-organize them into roles. Each role specified in `example.yml` should have its own directory under `roles/` and the tasks need be done by this role is listed in a file ‘`tasks/main.yml`’ as previous.

4.3.3 Using Variables

We demonstrate this feature by installing source code from Github. Although R can be installed through the OS package manager (`apt-get` etc.), the software used in your projects may not. Many research projects are available by Git instead. Here we are going to show you how to install packages from their Git repositories. Instead of directly executing the module ‘`apt`’, we pretend Ubuntu does not provide this package and you have to find it on Git. The source code of R can be found at <https://github.com/wch/r-source.git>. We are going to clone it to a remote VM’s hard drive, build the package and install the binary there.

To do so, we need a few new Ansible modules. You may remember from the last example that Ansible modules assist us to do different tasks based on the arguments we pass to it. It will come to no surprise that Ansible has a module ‘`git`’ to take care of git-related works, and a ‘`command`’ module to run shell commands. Let us modify `roles/r/tasks/main.yml` to be:

```
---
- name: get R package source
  git:
    repo: https://github.com/wch/r-source.git
    dest: /tmp/R

- name: build and install R
  become: yes
  command: chdir=/tmp/R "{{ item }}"
  with_items:
    - ./configure
    - make
```

```
- make install
```

The role `r` will now carry out two tasks. One to clone the R source code into `/tmp/R`, the other uses a series of shell commands to build and install the packages.

Note that the commands executed by the second task may not be available on a fresh VM image. But the point of this example is to show an alternative way to install packages, so we conveniently assume the conditions are all met.

To achieve this we are using variables in a separate file.

We typed several string constants in our Ansible scripts so far. In general, it is a good practice to give these values names and use them by referring to their names. This way, your complex Ansible project can be less error prone. Create a file in the same directory, and name it `vars.yml`:

```
---
repository: https://github.com/wch/r-source.git
tmp: /tmp/R
```

Accordingly, we will update our `example.yml`:

```
---
- hosts: R_hosts
  vars_files:
    - vars.yml
  roles:
    - r
```

As shown, we specify a `vars_files` telling the script that the file `vars.yml` is going to supply variable values, whose keys are denoted by Double curly brackets like in `roles/r/tasks/main.yml`:

```
---
- name: get R package source
  git:
    repo: "{{ repository }}"
    dest: "{{ tmp }}"

- name: build and install R
  become: yes
  command: chdir="{{ tmp }}" "{{ item }}"
  with_items:
    - ./configure
    - make
    - make install
```

Now, just edit the `hosts.txt` file with your target VMs' IP addresses and execute the playbook.

You should be able to extend the Ansible playbook for your needs.

Configuration tools like Ansible are important components to master the cloud environment.

4.3.4 Ansible Galaxy

Ansible Galaxy is a marketplace, where developers can share Ansible Roles to complete their system administration tasks. Roles exchanged in Ansible Galaxy community need to follow common conventions so that all participants know what to expect. We will illustrate details in this chapter.

It is good to follow the Ansible Galaxy standard during your development as much as possible.

4.3.4.1 Ansible Galaxy helloworld

Let us start with a simplest case: We will build an Ansible Galaxy project. This project will install the Emacs software package on your localhost as the target host. It is a *helloworld* project only meant to get us familiar with Ansible Galaxy project structures.

First you need to create a directory. Let us call it `mongodb`:

```
$ mkdir mongodb
```

Go ahead and create files `README.md`, `playbook.yml`, `inventory` and a subdirectory `roles/` then `playbook.yml` is your project playbook. It should perform the Emacs installation task by executing the corresponding role you will develop in the folder 'roles/'. The only difference is that we will construct the role with the help of `ansible-galaxy` this time.

Now, let `ansible-galaxy` initialize the directory structure for you:

```
$ cd roles
$ ansible-galaxy init <to-be-created-role-name>
```

The naming convention is to concatenate your name and the role name by a dot. [Figure 7](#) shows how it looks like.



Figure 7: image

Let us fill in information to our project. There are several `main.yml` files in different folders, and we will illustrate their usages.

defaults and vars:

These folders should hold variables key-value pairs for your playbook scripts. We will leave them empty in this example.

files:

This folder is for files need to be copied to the target hosts. Data files or configuration files can be specified if needed. We will leave it empty too.

templates:

Similar missions to files/, templates is allocated for template files. Keep empty for a simple Emacs installation.

handlers:

This is reserved for services running on target hosts. For example, to restart a service under certain circumstance.

tasks:

This file is the actual script for all tasks. You can use the role you built previously for Emacs installation here:

```
---
- name: install Emacs on Ubuntu 16.04
  become: yes
  package: name=emacs state=present
```

meta:

Provide necessary metadata for our Ansible Galaxy project for shipping:

```
---
galaxy_info:
  author: <you name>
  description: emacs installation on Ubuntu 16.04
  license:
    - MIT
```

```
min_ansible_version: 2.0
platforms:
  - name: Ubuntu
    versions:
      - xenial
galaxy_tags:
  - development

dependencies: []
```

Next let us test it out. You have your Ansible Galaxy role ready now. To test it as a user, go to your directory and edit the other two files `inventory.txt` and `playbook.yml`, which are already generated for you in directory `tests` by the script:

```
$ ansible-playbook -i ./hosts playbook.yml
```

After running this playbook, you should have Emacs installed on localhost.

4.3.5 A Complete Ansible Galaxy Project

We are going to use `ansible-galaxy` to setup a sample project. This sample project will:

- use a cloud cluster with multiple VMs
- deploy Apache Spark on this cluster
- install a particular HPC application
- prepare raw data for this cluster to process
- run the experiment and collect results

4.3.5.1 Ansible: Write a Playbooks for MongoDB

Ansible Playbooks are automated scripts written in YAML data format. Instead of using manual commands to setup multiple remote machines, you can utilize Ansible Playbooks to configure your entire systems. YAML syntax is easy to read and express the data structure of certain Ansible functions. You simply write some tasks, for example, installing software, configuring default settings, and starting the software, in a Ansible Playbook. With a few examples in this section, you will understand how it works and how to write your own Playbooks.

There are also several examples of using Ansible [Playbooks](#) from the official site. It covers

from basic usage of Ansible Playbooks to advanced usage such as applying

patches and updates with different roles and groups.

We are going to write a basic playbook of Ansible software. Keep in mind that `Ansible` is a main program and `playbook` is a template that you would like to use. You may have several playbooks in your Ansible.

4.3.5.2 First playbook for MongoDB Installation

As a first example, we are going to write a playbook which installs MongoDB server. It includes the following tasks:

- Import the public key used by the package management system
- Create a list file for MongoDB
- Reload local package database
- Install the MongoDB packages
- Start MongoDB

The material presented here is based on the manual installation of MongoDB from the official site:

- <http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/>*

We also assume that we install MongoDB on Ubuntu 15.10.

4.3.5.2.1 Enabling Root SSH Access

Some setups of managed nodes may not allow you to log in as root. As this may be problematic later, let us create a playbook to resolve this. Create a `enable-root-access.yaml` file with the following contents:

```
---
- hosts: ansible-test
  remote_user: ubuntu
  tasks:
    - name: Enable root login
      shell: sudo cp ~/.ssh/authorized_keys /root/.ssh/
```

Explanation:

- `hosts` specifies the name of a group of machines in the inventory
- `remote_user` specifies the username on the managed nodes to log in as

- `tasks` is a list of tasks to accomplish having a `name` (a description) and modules to execute. In this case we use the `shell` module.

We can run this playbook like so:

```
$ ansible-playbook -i inventory.txt -c ssh enable-root-access.yaml

PLAY [ansible-test] *****

GATHERING FACTS *****
ok: [10.23.2.105]
ok: [10.23.2.104]

TASK: [Enable root login] *****
changed: [10.23.2.104]
changed: [10.23.2.105]

PLAY RECAP *****
10.23.2.104      : ok=2    changed=1    unreachable=0    failed=0
10.23.2.105      : ok=2    changed=1    unreachable=0    failed=0
```

4.3.5.2.2 Hosts and Users

First step is choosing hosts to install MongoDB and a user account to run commands (tasks). We start with the following lines in the example filename of `mongodb.yaml`:

```
---
- hosts: ansible-test
  remote_user: root
  become: yes
```

In a previous section, we setup two machines with `ansible-test` group name. We use two machines for MongoDB installation. Also, we use `root` account to complete Ansible tasks.

Indentation is important in YAML format. Do not ignore spaces start

with in each line.

4.3.5.2.3 Tasks

A list of tasks contains commands or configurations to be executed on remote machines in a sequential order. Each task comes with a `name` and a `module` to run your command or configuration. You provide a description of your task in `name` section and choose a `module` for your task. There are several modules that you can use, for example, `shell` module simply executes a command without considering a return value. You may use `apt` or `yum` module which is one of the packaging modules to

install software. You can find an entire list of modules here: http://docs.ansible.com/list_of_all_modules.html

4.3.5.2.4 Module apt_key: add repository keys

We need to import the MongoDB public GPG Key. This is going to be a first task in our playbook.:

```
tasks:
- name: Import the public key used by the package management system
  apt_key: keyserver=hkp://keyserver.ubuntu.com:80 id=7F0CEB10 state=present
```

4.3.5.2.5 Module apt_repository: add repositories

Next add the MongoDB repository to apt:

```
- name: Add MongoDB repository
  apt_repository: repo='deb http://downloads-distrow.mongodb.org/repo/ubuntu-upstart dist 10gen' state=present
```

4.3.5.2.6 Module apt: install packages

We use `apt` module to install `mongodb-org` package. `notify` action is added to start `mongod` after the completion of this task. Use the `update_cache=yes` option to reload the local package database.:

```
- name: install mongodb
  apt: pkg=mongodb-org state=latest update_cache=yes
  notify:
  - start mongodb
```

4.3.5.2.7 Module service: manage services

We use `handlers` here to start or restart services. It is similar to `tasks` but will run only once.:

```
handlers:
- name: start mongodb
  service: name=mongod state=started
```

4.3.5.2.8 The Full Playbook

Our first playbook looks like this:

```
---
- hosts: ansible-test
  remote_user: root
  become: yes
  tasks:
```

```

- name: Import the public key used by the package management system
  apt_key: keyserver=hkp://keyserver.ubuntu.com:80 id=7F0CEB10 state=present
- name: Add MongoDB repository
  apt_repository: repo='deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' state=present
- name: install mongodb
  apt: pkg=mongodb-org state=latest update_cache=yes
  notify:
    - start mongodb
handlers:
  - name: start mongodb
    service: name=mongod state=started

```

4.3.5.2.9 Running a Playbook

We use `ansible-playbook` command to run our playbook:

```

$ ansible-playbook -i inventory.txt -c ssh mongodb.yaml

PLAY [ansible-test] *****

GATHERING FACTS *****
ok: [10.23.2.104]
ok: [10.23.2.105]

TASK: [Import the public key used by the package management system] *****
changed: [10.23.2.104]
changed: [10.23.2.105]

TASK: [Add MongoDB repository] *****
changed: [10.23.2.104]
changed: [10.23.2.105]

TASK: [install mongodb] *****
changed: [10.23.2.104]
changed: [10.23.2.105]

NOTIFIED: [start mongodb] *****
ok: [10.23.2.105]
ok: [10.23.2.104]

PLAY RECAP *****
10.23.2.104      : ok=5    changed=3    unreachable=0    failed=0
10.23.2.105      : ok=5    changed=3    unreachable=0    failed=0

```

If you rerun the playbook, you should see that nothing changed:

```

$ ansible-playbook -i inventory.txt -c ssh mongodb.yaml

PLAY [ansible-test] *****

GATHERING FACTS *****
ok: [10.23.2.105]
ok: [10.23.2.104]

TASK: [Import the public key used by the package management system] *****
ok: [10.23.2.104]
ok: [10.23.2.105]

TASK: [Add MongoDB repository] *****
ok: [10.23.2.104]
ok: [10.23.2.105]

TASK: [install mongodb] *****
ok: [10.23.2.105]
ok: [10.23.2.104]

PLAY RECAP *****
10.23.2.104      : ok=4    changed=0    unreachable=0    failed=0
10.23.2.105      : ok=4    changed=0    unreachable=0    failed=0

```

4.3.5.2.10 Sanity Check: Test MongoDB

Let us try to run ‘mongo’ to enter mongodb shell.:

```
$ ssh ubuntu@$IP
$ mongo
MongoDB shell version: 2.6.9
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
>
```

4.3.5.2.11 Terms

- **Module:** Ansible library to run or manage services, packages, files or commands.
- **Handler:** A task for notifier.
- **Task:** Ansible job to run a command, check files, or update configurations.
- **Playbook:** a list of tasks for Ansible nodes. YAML format used.
- **YAML:** Human readable generic data serialization.

4.3.5.2.12 Reference

The main tutorial from Ansible is here:
http://docs.ansible.com/playbooks_intro.html

You can also find an index of the ansible modules here:
http://docs.ansible.com/modules_by_category.html

4.3.6 Exercise

We have shown a couple of examples of using Ansible tools. Before you apply it in you final project, we will practice it in this exercise.

- set up the project structure similar to Ansible Galaxy example
- install MongoDB from the package manager (apt in this class)
- configure your MongoDB installation to start the service automatically

- use default port and let it serve local client connections only

5 DRAFTS

5.1 DATA SERVICES

5.1.1 Box



Learning Objectives

- Learn about Box
 - Use python to interface with Box
-

Box is cloud storage service that allows users to store, access, collaborate, and share files, similar to DropBox. However, while DropBox started out as a service for storing personal files, Box is geared more towards business applications. Box also has its own platform offering APIs in multiple languages and an SDK for the development of custom applications and integrations, as well as many pre-built apps for integrating Box into various other tools and platforms. It has some limited project management tools in addition to its storage capabilities, including task and workflow management. Box offers free and paid versions for individual accounts and multiple types of business accounts that are charged on a per user basis. Finally, Box has just released Box Skills, a machine learning tool for automatically processing files uploaded to Box.

5.1.1.1 Limitations

While Box offers unlimited storage, it's biggest business account has a 5GB individual file size limit with a 2GB limit on the smallest business plan and 250MB on the unpaid personal plan. Other services have no individual file size limit. Using Box Sync grants the user full access to the data in the sync, including the ability to delete the data. Restoring data yourself only restores flat-folders and not nested ones, in order to fully restore everything Box must do the restoration. Problems can occur when two users edit the same file at the same time, unlike other collaboration tools. While there is no official limit on the

number of files uploaded at one time, Box itself recommends users not exceed 100,000 files at a time. Deleting a user's account also deletes all the information they own, which can be problematic for users leaving a company. While Box has a collections feature, the only collection supported is the `Favorites` collection, users are not able to make their own

REST:

- [Box developer webpage](#)

Installation:

```
$ pip install boxsdk
```

If you will be using JWT authentication for your app, you will want to install its dependencies:

```
$ pip install "boxsdk[JWT]"
```

5.1.1.2 Creating an app

Once you have created a Box account, go to the Developer Console and select `Create New App`. You will need to select what type of application you are building and an authentication method for your app and then enter an app name (you can change this later). Once your app has been created, click `View App`. You will then need to click the profile button in the top right corner of the page, and go to `Account Settings`. Scroll down to the Authentication section and click `Require 2-step verification for unrecognized logins`, then follow the prompts.

The following examples have been adapted from <https://developer.box.com/reference>.

5.1.1.2.1 Authentication with JWT

In the Configuration panel of the Developer Console, scroll down to the section titled `Add and Manage Public Keys` and click `Generate a Public/Private Keypair`:

Add and Manage Public Keys

Generate an RSA keypair to sign and authenticate the JWT request made by your app or upload your own public key.

[Learn to generate your own RSA keypair.](#)

Note: Box does not store your private key, so make sure you save the downloaded file if you are using our generate button.

Add a Public Key

Generate a Public/Private Keypair

Box Add Key

Once you have generated a keypair, a config.json file will automatically download. Save this file in a secure location as you will need it for authentication purposes. Finally, you will need to read in this config file into your app:

```
from boxsdk import JWTAuth
from boxsdk import Client

sdk = JWTAuth.from_settings_file('<path to config.json>')
client = Client(sdk)
```

For OAuth 2 authentication see <https://developer.box.com/docs/authenticate-with-oauth-2>.

5.1.1.3 Box Methods

The Python SDK has several methods for creating objects and endpoints which you can then perform operations on.

5.1.1.3.1 Get information about a Box object

A call to `get()` will return general information about a Box object, including id, name, and other object specific information.

```
# Get information about the logged in user
# (that is whoever owns the developer token):
user = client.user().get()
print(user.name)
print(user.login)
print(user.avatar_url)
```

The root directory will always have `'0'` as the id:

```
# Get information about the root folder (referenced by id '0'):
folder = client.folder('0').get()
print(folder.name)
print(folder.item_status)
```

5.1.1.3.2 Folders

Folders can hold other folders as well as files.

```
# Create a new folder:
subfolder = client.folder('<folder id>').create_subfolder('<subfolder name>')

# Delete a folder:
client.folder('<folder id>').delete()
```

You can also copy existing folders or update a current folder.

```
# Copy a folder:
folder = client.folder('<folder id>')
destination = client.folder('<destination folder id>')
copy_of_folder = folder.copy(destination)

# Update a folder:
folder = client.folder('<folder id>').update_info({'name': 'Updated name', 'description': 'This has now been updated.'})
```

Calling `get_items()` will return all items in a folder, including files and sub-folders.

```
# Get all items in a folder:
items = client.folder('<folder id>').get_items()
for item in items:
    print(item.id)
```

5.1.1.3.3 Uploading files

Files can be uploaded from local files or from stream objects. Updating a file is as simple as editing a local copy then calling `update_contents` on the file on Box.

```
# Upload a file to a Box folder:
test_file = client.folder('<folder id>').upload('<file path>')
print(test_file.name)

# Upload a stream to a Box folder:
from io import StringIO
stream = StringIO()
stream.write("Test stream")
stream.seek(0)
stream_file = client.folder('0').upload_stream(stream, 'Stream File')
print(stream_file.name)
print(stream_file.content())

# Upload a new version of a file:
client.file('<file id>').update_contents('<path to file>')
```

5.1.1.3.4 File upload errors

A file upload will fail if there is already a file in the folder with the same name, if the file is too big, or if there is not enough storage. To avoid errors, Box has an

exception API that will check if a file will be accepted before sending it to Box:

```
# Enable preflight checks:
file = 'test.txt'
try:
    test_file = client.folder('0').upload('test.txt', 'Test File', preflight_check=True)
    print(test_file.name)
    print(test_file.content())
except BoxAPIException:
    pass
```

Which will return the following:

```
"OPTIONS https://api.box.com/2.0/files/content" 409 466
{'Date': 'Thu, 24 Jan 2019 16:30:46 GMT',
 'Content-Type': 'application/json',
 'Transfer-Encoding': 'chunked',
 'Connection': 'keep-alive',
 'Strict-Transport-Security': 'max-age=31536000',
 'Cache-Control': 'no-cache, no-store',
 'Content-Encoding': 'gzip',
 'Vary': 'Accept-Encoding',
 'BOX-REQUEST-ID': '0rgjouev2logn8gqcn1fauco84o',
 'Age': '0'}
{'code': '---use',
 'context_info': {
   'conflicts': {
     'etag': '0',
     'file_version': {'id': '411411432162',
                      'sha1': '02d92c580d4ede6c80a878bdd9f3142d8f757be8',
                      'type': 'file_version'},
     'id': '389113382562',
     'name': 'Test File',
     'sequence_id': '0',
     'sha1': '02d92c580d4ede6c80a878bdd9f3142d8f757be8',
     'type': 'file'}}},
 'help_url': 'http://developers.box.com/docs/#errors',
 'message': 'Item with the same name already exists',
 'request_id': 'slj5jkkfzi55kba81',
 'status': 409,
 'type': 'error'}
```

5.1.1.3.5 Deleting, copying, and downloading files

Individual files can be downloaded by specifying the name of the output file.

```
# Download a file:
with open('<output file name>', 'wb') as f:
    client.file("<file id>").download_to(f)
```

Deleting and copying files is similar to deleting and copying folders.

```
# Delete a file:
client.file('<file id>').delete()

# Copy a file:
file = client.file('<file id>')
destination = client.folder('<folder id>')
copy_of_file = file.copy(destination)
```

5.1.1.3.6 Searching

The query string used in a search can include object names, description, text

content, or other object data.

```
items = client.search().query('<query string>', file_extensions = ['png', 'txt'], fields = ['name', 'description'])
for item in items:
    print(item.name)
```

5.1.1.3.7 Shared links

Shared links give read-only access to a file through a URL. Specifying the access level of a shared link determines whether users will need to authenticate with Box in order to view the file.

```
# Creating a shared link:
url = client.file('<file id>').get_shared_link()

# Retrieving a shared link that has already been created:
url = client.file('<file id>').shared_link['url']
```

5.1.1.4 Project management

Box offers some limited project management tools, including groups, collaborations, and tasks. Note: you can create another user to test out project management tools as follows:

```
test_user = client.create_user('test user', login=None)
print(user.id)
```

5.1.1.4.1 Collaborations

A collaboration object gives a user specified permissions for the defined files and folders. The collaboration object itself returns information about the users, files, and roles of the collaboration.

```
collaboration = client.collaboration('<collab id>').get()
print(collab.role)
print(collab.item['type'])
```

Roles include editor, viewer, previewer, uploader, previewer uploader, viewer uploader, or co-owner.

```
# Create a new collaboration
from boxsdk.object.collaboration import CollaborationRole
user = client.user(user_id='<user id>')
collab = client.folder(folder_id='<folder id>').collaborate(user, CollaborationRole.VIEWER)
```

Updating and deleting a collaboration is similar to other box objects.

```
# Update a collaboration
from boxsdk.object.collaboration import CollaborationRole
```

```

collaboration = client.collaboration(collab_id='<collaboration id>')
updated_collaboration = collaboration.update_info(CollaborationRole.EDITOR)

# Delete a collaboration
client.collaboration(collab_id='<collaboraiton id>').delete()

```

5.1.1.4.2 Groups

A group object can be used instead of a user in collaborations. The `get()` call to a group object returns basic information about the group and does not include a member list.

```

# Create, update, or delete a group
new_group = client.create_group(name="New Group")
updated_group = client.group(group_id='<group id>').update_info({'key': 'value'})
client.group(group_id='<group id>').delete()

# Add a member to a group
user = client.user(user_id='<user id>')
member = client.group(group_id='<group id>').add_member(user)

group = client.group(group_id='<group id>').get()
{
  "type": "group",
  "id": "255224",
  "name": "Everyone",
  "created_at": "2014-09-15T13:15:35-07:00",
  "modified_at": "2014-09-15T13:15:35-07:00"
}

```

To get all members in a group, you must call `get_memberships()`:

```

memberships = client.group(group_id = '<group id>').get_memberships()
for member in memberships:
    print(member.user['id'])

```

Once a member has been added to the group, a membership object is created which controls the relationship after creation.

```

# Get, update, and delete membership:
membership = client.group_membership('<membership id>').get()
print(membership.user.name)
updated_membership = client.group_membership('<membership id>').update_info({'key': 'value'})
client.group_membership('<membership id>').delete()

```

You can see all groups a user is a member of with by calling `get_group_memberships` on a user object:

```

memberships = client.user(user_id='<user id>').get_group_memberships()
for m in memberships:
    print(m.group.name)

```

You can see all collaboration objects a group has by calling `get_collaborations` on a group object:

```

collaborations = client.group('<group id>').get_collaborations()
for collab in collaborations:
    print(collab.item.name)

```

5.1.1.4.3 Tasks

Tasks are attached to file objects and can be assigned to specific users by creating a task assignment.

```
# Create a new task
task = client.file(file_id='<file id>').create_task('<due date>')
```

Tasks can be updated, deleted with calls to `update()` and `delete()`. Calling `get()` will return general info about the task. Tasks can be assigned to a user by calling `assign()` which will create a task assignment object.

```
user = client.user(user_id='<user id>')
assignment = client.task('<task id>').assign(user)
print(assignment.id)
```

5.1.1.5 Pybox

Pybox provides a way to work with Box files from the command line. Documentation on how to set up and use pybox can be found at <https://github.com/hzheng/pybox>

5.1.1.6 Cloudmesh



Box file storage can now be used from within the Cloudmesh library. Once you have created your config file, you must add the path to your cloudmesh yaml file under box credentials. The Cloudmesh command line library offers six functions for file storage: get, put, search, list, create directory, and delete. Once you have installed Cloudmesh, type `cms` into the command line to start the shell. Every box command should start with the following:

```
$ storage --storage=box
```

5.1.1.6.1 Get

To download a file from Box with Cloudmesh, you must specify the cloud folder or file to be downloaded and the local folder to download to. To download all the contents of a folder, simply specify a folder on the cloud and use the recursive option.


```
$ storage --storage=box get /test_folder/test_file.txt ~/test_folder --recursive
```

5.1.1.6.2 Put

The put command uploads files from your local host to the cloud. If you specify a file as the source, the file will be uploaded if no such file exists on the cloud or updated if a copy already exists on the cloud. If the source is a directory and recursive is specified, Cloudmesh will upload all the contents of the source directory to the cloud.

```
$ storage --storage=box put ~/test_folder /uploads --recursive
```

5.1.1.6.3 Search

To search for a file through Cloudmesh, you must specify a directory in which to search and the file or folder name you are searching for. If recursive is specified, Cloudmesh will search all child directories of the original directory.

```
$ storage --storage=box search /uploads last_upload.txt --recursive
```

5.1.1.6.4 List

The list command lists all the contents of a cloud directory. If recursive is specified, it will list the contents of all child directories as well.

```
$ storage --storage=box list /uploads --recursive
```

5.1.1.6.5 Create a directory

To create a new directory, you must specify the path of the new directory you would like to create, including its parent directory.

```
$ storage --storage=box create dir /test_folder/new_folder
```

5.1.1.6.6 Delete

The delete command can delete files or folders from your cloud file storage. Deleting a folder will delete its contents as well.

```
$ storage --storage=box delete /uploads/last_upload.txt
```

Finally, if you have set the storage variable to box, you can omit the `--storage=box`

from your command line calls.

 openapi

5.1.2 AWS DocumentDB (hid: sp19-516-134)



Learning Objectives

- Learn about AWS Document DB
 - Components
 - Features and Benefits
 - Pricing
 - How to provision Document database
 - How to connect to Document Database
-

AWS Document DB is a fully managed service with compatibility with Mongo DB. Mongo DB application code can be run here with same drivers and tools. A cluster needs to be created for using AWS document DB. It can contain zero or more databases instances and a attached volume for managing data for the cluster. The storage is replicated to multiple availablity zones with each having its own copy.

Components:

1. Instances - There could be upto 16 instances which do the reading and writing data from storage volume. Primary and Replica are the two types of instances. Only one Primary instances is permitted which is used to read and write to the volume. Replica is used only for read operation and can be used to placed in multiple availbaility zones to increase the cluster availability. Instances can be bought up and terminated as desired. Compute capacity can be scaled independent of storage.
2. Cluster Volumes - Volume can store upto 64 TB of data replicated across availability zones.

5.1.2.1 Amazon DocumentDB Features and Benefits

- *MongoDB-compatible*: Implemented using the Apache 2.0 open source MongoDB 3.6 API by emulating the responses that a MongoDB client expects from a MongoDB server. This allows you to use your existing MongoDB drivers and tools with Amazon DocumentDB.
- *Highly available*: Designed for 99.99% availability with six copies of data across three AWS Availability Zones (AZs). Switches to read replica in the event of a failure—typically in less than 30 seconds. Data is automatically backed up in S3 for 35 days.
- *Performance at scale*: It uses a distributed, fault-tolerant, self-healing storage system that auto-scales up to 64 TB per database cluster. Reduces database I/O by writing only database changes to the storage layer
- *Highly secure*: Provides multiple levels of security for your database, including network isolation using Amazon VPC, encryption at rest using keys you create and control through AWS Key Management Service (KMS), and encryption-in-transit using Transport Layer Security (TLS).
- *Fully managed*: Automatically and continuously monitors and backs up your database to Amazon S3, enabling point-in-time recovery (up to the second for the last 35 days). Integrates with Amazon CloudWatch, so you can monitor over 20 key operational metrics for your database instances via the AWS Management Console.

5.1.2.2 AWS Document database Pricing

- *On-Demand Instance Pricing*: No commitments and pay by hour
- *Database Storage and IOs* : Storage is billed in per GB-month increments and IOs are billed in per million request increments.
- *Backup Storage* : Cost associated with cluster backups and any customer-initiated cluster snapshots.

5.1.2.3 How to provision Document database

User needs to have a AWS account to use the Document DB service. First step is to create a AWS account. Once created login to the account.

An AWS account can be created using the link below

[AWS account creation URL](#)

5.1.2.3.1 Step 1: Login to the AWS console.

Login to the AWS account using the link below.

[AWS Console URL](#).

Upon successful login, select Document DB from the Database section or alternatively, you can type DocumentDB in the search bar to look up.

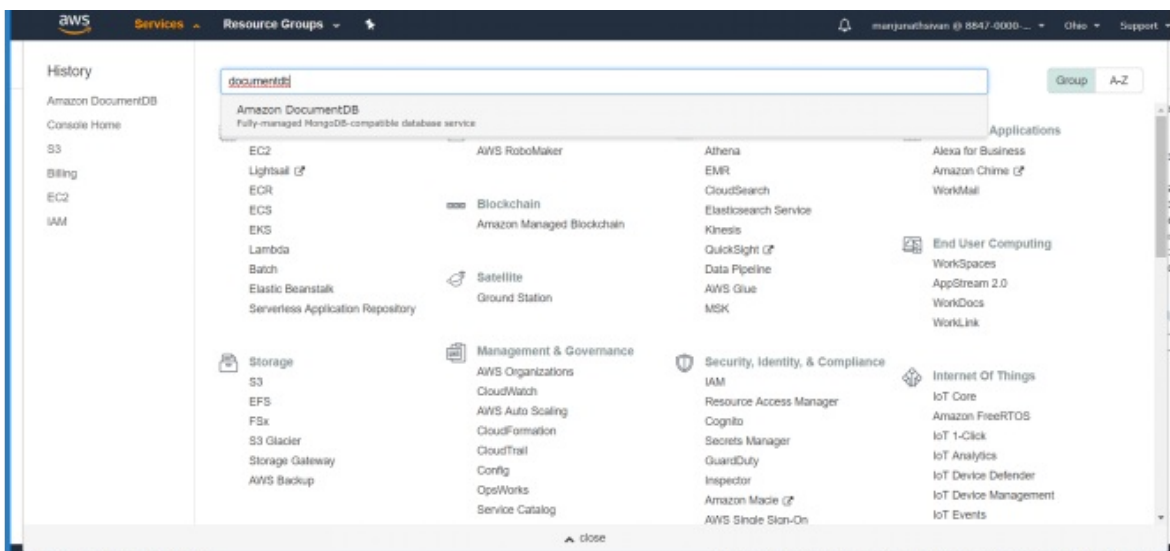


Figure 8: AWS DocumentDB

5.1.2.3.2 Step 2: Click on Create Amazon DocumentDB cluster button.

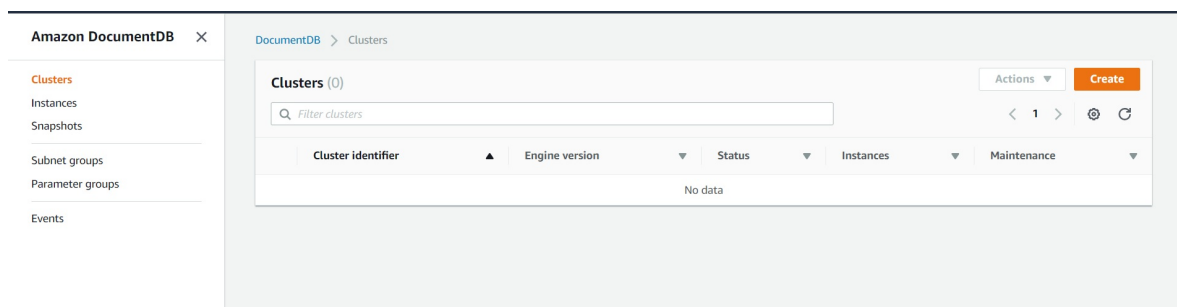
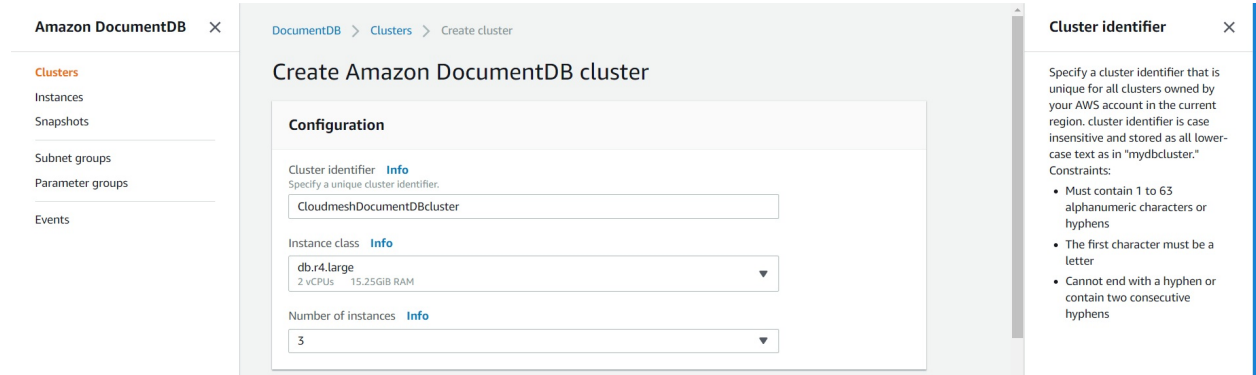


Figure 9: AWS DocumentDB

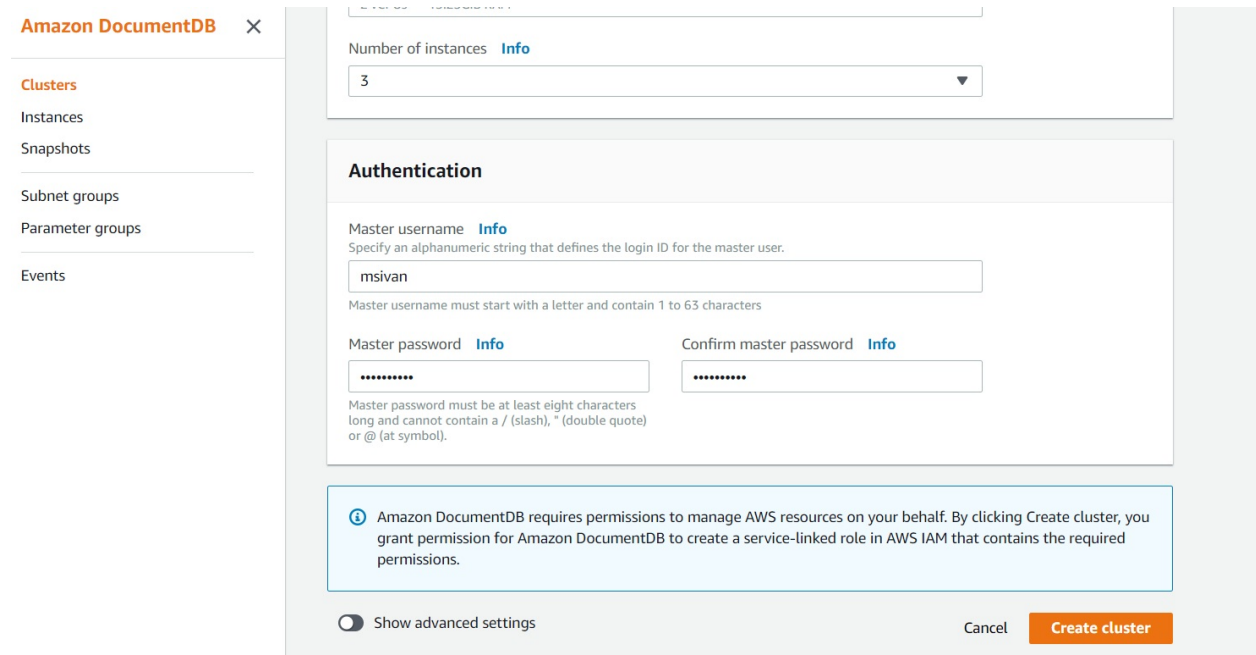
5.1.2.3.3 Step 3: Create the required configuration.

1. Specify a unique cluster identifier
2. Specify the compute and memory capacity of the instance.
3. Select the number of instances to be deployed in your cluster.



{#fig:aws-create cluster page}

Enter Authentication information if required.



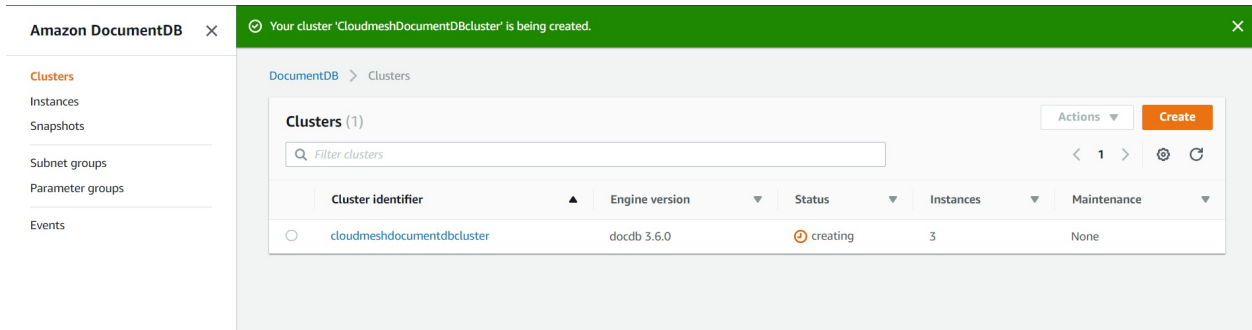
{#fig:aws-create cluster authentication section}

Once done click the create cluster button

5.1.2.3.4 Step 3: Cluster creation process

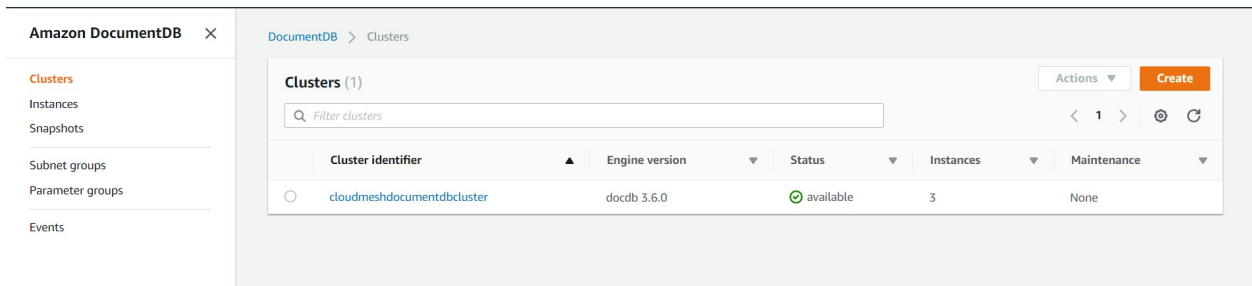
Once all the required configuration has been entered and create cluster is clicked. Document DB creates the clusters with the number of instance that was provided as part of cluster configuration. It takes sometime depending on the number of instances in the cluster.

You will see a Cluster being created message



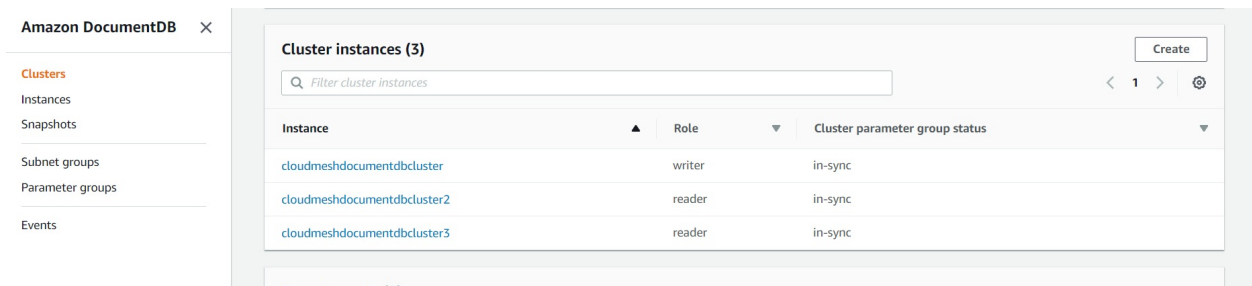
{#fig:aws-cluster creation}

Once the cluster is created , you can see the status as available.



{#fig:aws-cluster available}

All the cluster instances that got created in the process as per configuration can be seen in the details.



{#fig:aws-cluster Instances}

5.1.2.3.5 Step 4: Cluster summary and connection information from Mongo and Application.

Cluster Information along with the required connection information from Mongo and for application is available to connect.

The screenshot shows the Amazon DocumentDB console interface. On the left is a navigation sidebar with options: Clusters (highlighted), Instances, Snapshots, Subnet groups, Parameter groups, and Events. The main content area is titled 'Summary' and contains a table with the following data:

Engine version	Cluster status	Pending maintenance	Cluster parameter group status
docdb 3.6.0	available	None	in-sync

Below the summary is a 'Connect' section with two code blocks:

```
Connect to this cluster with the mongo shell  
mongo --ssl --host ccloudmeshdocumentdbcluster.cluster-centkgpnq9qx.us-east-2.docdb.amazonaws.com:27017 --sslCAFile rds-combined-ca-bundle.pem --username msivan --password <insertYourPassword>
```

```
Connect to this cluster with an application  
mongodb://msivan:<insertYourPassword>@ccloudmeshdocumentdbcluster.cluster-centkgpnq9qx.us-east-2.docdb.amazonaws.com:27017/?ssl=true&ssl_ca_certs=rds-combined-ca-bundle.pem&replicaSet=rs0
```

{#fig:aws-cluster connection Information}

5.1.2.3.6 Step 5: Cluster Details.

Cluster configuration and status along with backup , maintenance details and security network information can be viewed in cluster details section

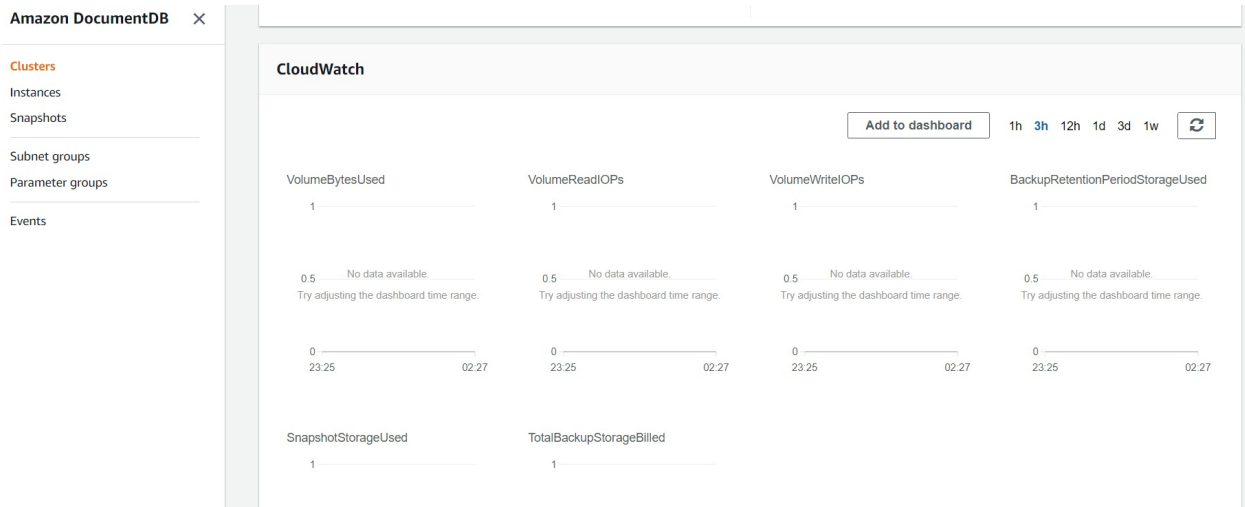
The screenshot shows the Amazon DocumentDB console 'Cluster details' page. The left sidebar is the same as in the previous screenshot. The main content area is titled 'Cluster details' and has a 'Modify' button in the top right corner. It is divided into three sections:

- Configurations and status:**
 - ARN: `arn:aws:rds:us-east-2:884700007966:cluster:cloudmeshdocumentdbcluster`
 - Cluster identifier: `cloudmeshdocumentdbcluster (available)`
 - Cluster creation time: `3/26/2019, 10:25:52 PM UTC-4`
 - Cluster endpoint: `cloudmeshdocumentdbcluster.cluster-centkgpnq9qx.us-east-2.docdb.amazonaws.com`
 - Reader endpoint: `cloudmeshdocumentdbcluster.cluster-ro-centkgpnq9qx.us-east-2.docdb.amazonaws.com`
 - Master username: `msivan`
 - Port: `27017`
 - Status: (partially visible)
- Backup:**
 - Automated backups: `Enabled (1 day)`
 - Earliest restorable time: `3/26/2019, 10:26:49 PM UTC-4`
 - Latest restore time: `3/26/2019, 10:26:49 PM UTC-4`
 - Backup window: `00:00-00:30 UTC (GMT)`
- Maintenance details:**
 - Maintenance window: `thu:03:11-thu:03:41 UTC (GMT)`
- Security and network:**
 - Encryption-at-rest: `Yes`
 - KMS key: (partially visible)

{#fig:aws-cluster details}

5.1.2.3.7 Step 6: Cloudwatch Information on the clusters.

Cloudwatch provides differnt type of metrics to keep a tab on the clusters



{#fig:aws-cluster cloudwatch}

5.1.2.4 How to connect to Document Database

TLS is enabled on Amazon DocumentDB clusters by default. We can connect to Document DB in both cases where in the TLS is enabled or not. To connect to Document DB with TLS enabled , following steps needs to be taken programatically in python

```
import pymongo
import sys

# Create a MongoDB client and open connection to Amazon DocumentDB
client = pymongo.MongoClient('mongodb://<dbusername>:<dbpassword>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?ssl')

# Specify the database to be used
db = client.test

# Specify the collection to be used
col = db.myTestCollection

# Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

# Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

# Print the result to the screen
print(x)

# Close the connection
client.close()
```

Similarly to connect to the cluster with TLS disabled, following code can be used.

```
# Create a MongoDB client and open connection to Amazon DocumentDB
client = pymongo.MongoClient('mongodb://<dbusername>:<dbpassword>@mycluster.node.us-east-1.docdb.amazonaws.com:27017/?rep
```



```
# Specify the database to be used
db = client.test

# Specify the collection to be used
col = db.myTestCollection

# Insert a single document
col.insert_one({'hello':'Amazon DocumentDB'})

# Find the document that was previously written
x = col.find_one({'hello':'Amazon DocumentDB'})

# Print the result to the screen
print(x)

# Close the connection
client.close()
```

5.1.3 Amazon Aurora DB



Learning Objectives

- Learn about AWS Aurora DB
 - Features and Benefits
 - Pricing
 - How to provision Aurora database
 - How to connect to Aurora Database
-

Amazon Aurora is an on-cloud relational database offering from AWS that aims to offer the performance and reliability of traditional enterprise databases combined with cost-effectiveness of open source databases.

Aurora comes as an SaaS offering in RDS suite offered by Amazon.

5.1.3.1 Amazon Aurora Features and Benefits

The following features and benefits are highlighted by Amazon:

- *Compatible Databases:* Currently, Aurora is compatible with MySQL and PostgreSQL Databases. It provides a wrapper to provision these open source databases and manage them for high availability. This allows users to provision these open source databases through Amazon and still use

existing code, tools and drivers with little change.

- *High Performance:* Aurora offers up to 5 times the throughput of a standard MySQL database and 3 times throughput of standard PostgreSQL. All this is offered at a price point which is 1/10th of a commercial database.
- *Scalability:* Aurora allows users to scale up and down the databases to smaller or larger sized servers based on the dynamic business needs to match the required compute power. There is also a serverless offering where AWS manages the scaling of compute requirements. Aurora also adds additional storage as needed up to 64TB per instance as the data grows.
- *High Availability and Durability:* Aurora DB offers multi AZ option to make the data replicated across more than one availability zone making it resilient to failures. Data can be backed up to Amazon S3 to enable point in time recovery in case of errors.
- *High Performance through Read Replicas:* Aurora DB offers to create up to 15 low latency Read Replica nodes for a database which allows for high performance. One can have a writer node to write data into the database and use the Read Replica nodes for query data.
- *Fully Managed:* Aurora comes as part of the RDS suite where Amazon manages the database management activities like hardware provisioning, set up and configuration, software patching, database backups and performance monitoring.
- *Security:* Aurora allows to secure the data at rest and in transit by using keys through Key Management Service. Databases can also be made part of a VPC and secured using private subnets and security groups. Database encryption can be enabled at the time of database creation which means that the data, backups, snapshots and replicas are also encrypted.
- *Parallel Query for Analytical Queries:* This feature allows users to run analytical queries on the database without the need to copy the data into another system to not impact the system performance. Aurora offloads the query to the CPU nodes in its storage layer allowing transactional and analytical loads alongside each other.
- *Performance Monitoring:* Aurora provides Performance Insight as a database performance tuning and monitoring feature that can be enabled on the database for additional cost. It allows to visualize the loads to identify performance issues.
- *Support for Migration:* Amazon provides tools for migrating existing

MySQL and PostgreSQL to Aurora. AWS Database Migration Service is provided as a service to migrate from commercial databases into Aurora.

5.1.3.2 Amazon Aurora Pricing

Amazon Aurora pricing is determined by various factors:

- Type and Size of database instance
- Storage and IO
- Backup Storage
- Data Transfer

For detailed pricing refer AWS Aurora Pricing documentation [AWS Aurora Pricing](#).

5.1.3.3 How to provision Aurora database

In this section we will discuss the steps to be followed to provision a database using AWS Aurora service.

To be able to use AWS Aurora service, the user must have set up an AWS account as a pre-requisite. An AWS account can be create using the link below

[New AWS account creation URL](#)

5.1.3.3.1 Step 1: Login to the AWS console.

Login to the AWS account using the link below.

[AWS Console URL](#).

Upon successful login, select RDS from the Database section or alternatively, you can type RDS in the search bar to look up (see [Figure 10](#)).

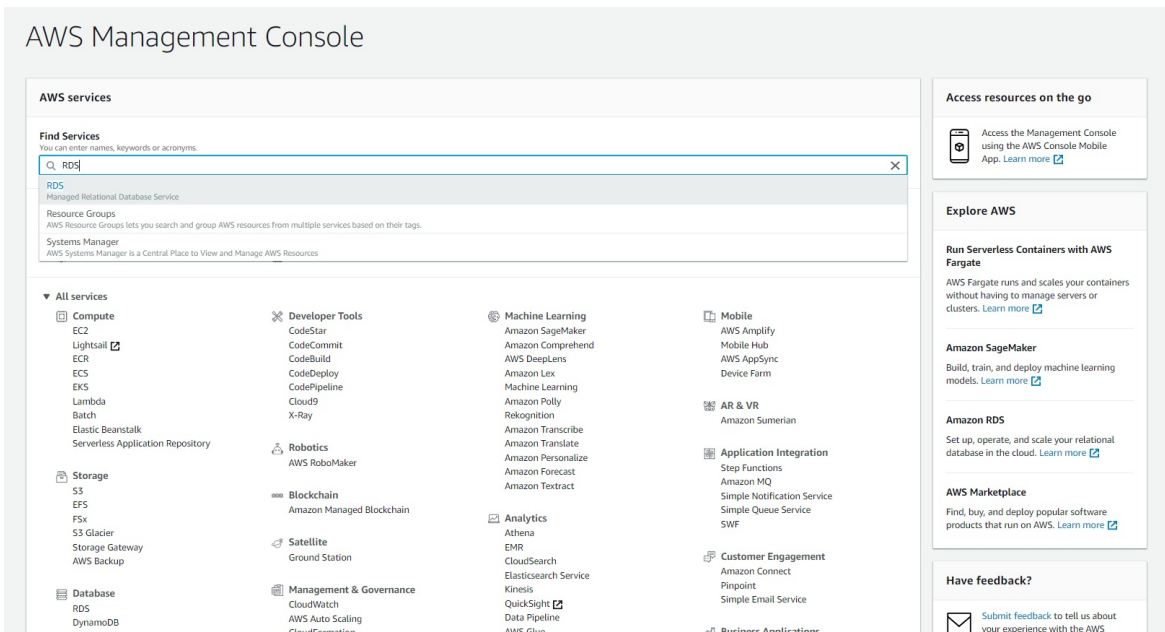


Figure 10: AWS Aurora DB

5.1.3.3.2 Step 2: Click on Create Database

On the RDS home page, one can either click on the `create Database` button in the Amazon Aurora section directly or click on the `create Database` button in the Create Database section (see [Figure 11](#)).

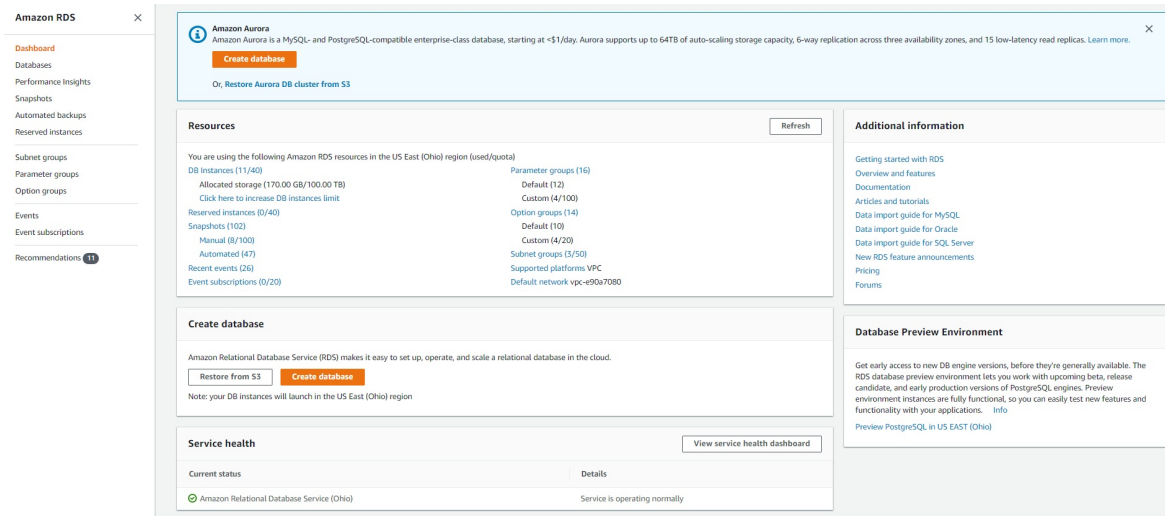


Figure 11: AWS Aurora DB

5.1.3.3.3 Step 3: Select Amazon Aurora Engine

Create Database section lists all RDS database flavors available. Choose

Amazon Aurora database on the list. After choosing Amazon Aurora, AWS will display list of available editions within MySQL and PostgreSQL engines. At the time of writing this page, MySQL 5.6-compatible is comes with Serverless and Parallel Query capabilities, hence we will go with this option (see [Figure 12](#)).

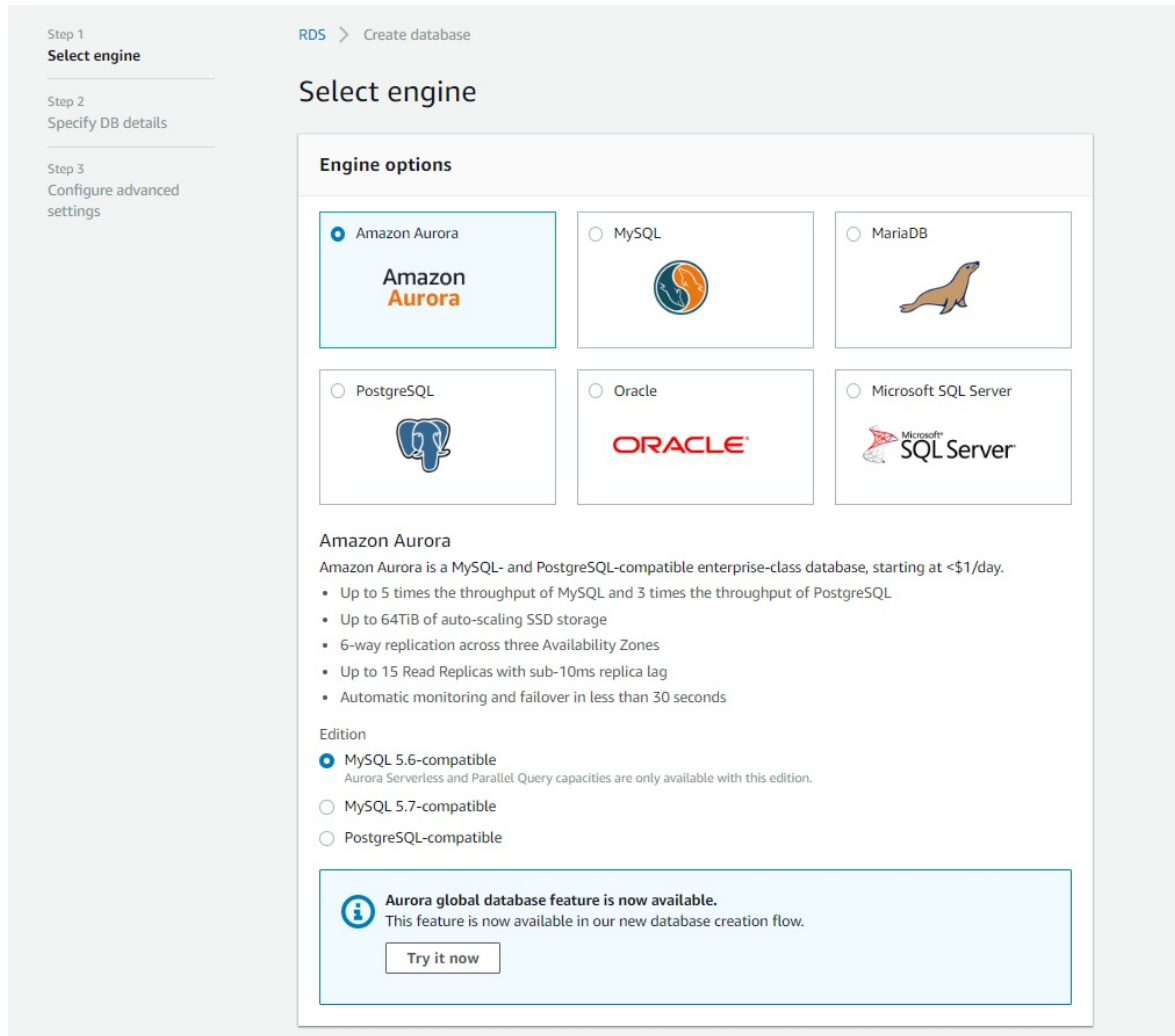


Figure 12: AWS Aurora DB

5.1.3.3.4 Step 4: Aurora Configuration and Settings

Once we select the engine, next step is to select the Configuration and Setting (see [Figure 13](#)).

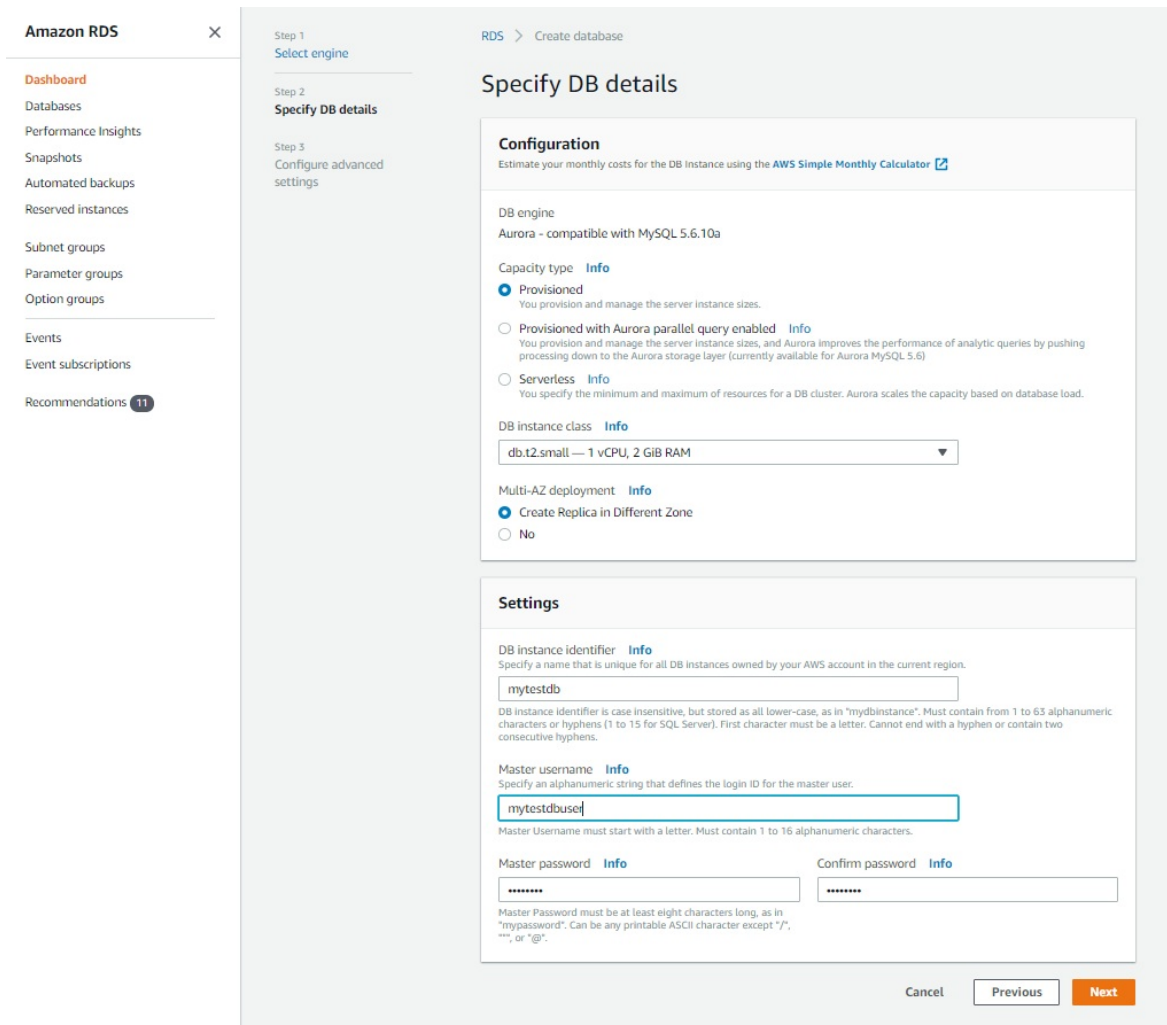


Figure 13: AWS Aurora DB

5.1.3.3.4.1 Configuration

Capacity Type: This attribute allows us to select between Provisioned, Provisioned (with Parallel Query Feature) and Serverless offerings. We will go with provisioned option in this example.

DB Instance Class: This attribute allows to select the type of instance based on the need like small, medium or large. It defines the number of CPUs and RAM available to the DB server.

Multi AZ - Deployment: Selecting this option means that Aurora will create a Read Replica in the a different availability zone. This can be used as a reader node for faster read performance and as a standby by in case the primary DB

server becomes unavailable due to some issue.

5.1.3.3.4.2 Settings

DB Instance Identifier: This will be a unique name that will be used for the writer DB node.

Master UserName and Password: These attribute define the master user and password for the DB that can be used to manage the database once created.

5.1.3.3.4.3 Step 5: Configure Advance Settings - Network and Security

This page will help configure some additional settings for the database. We will cover them in the next few steps (see [Figure 14](#)).

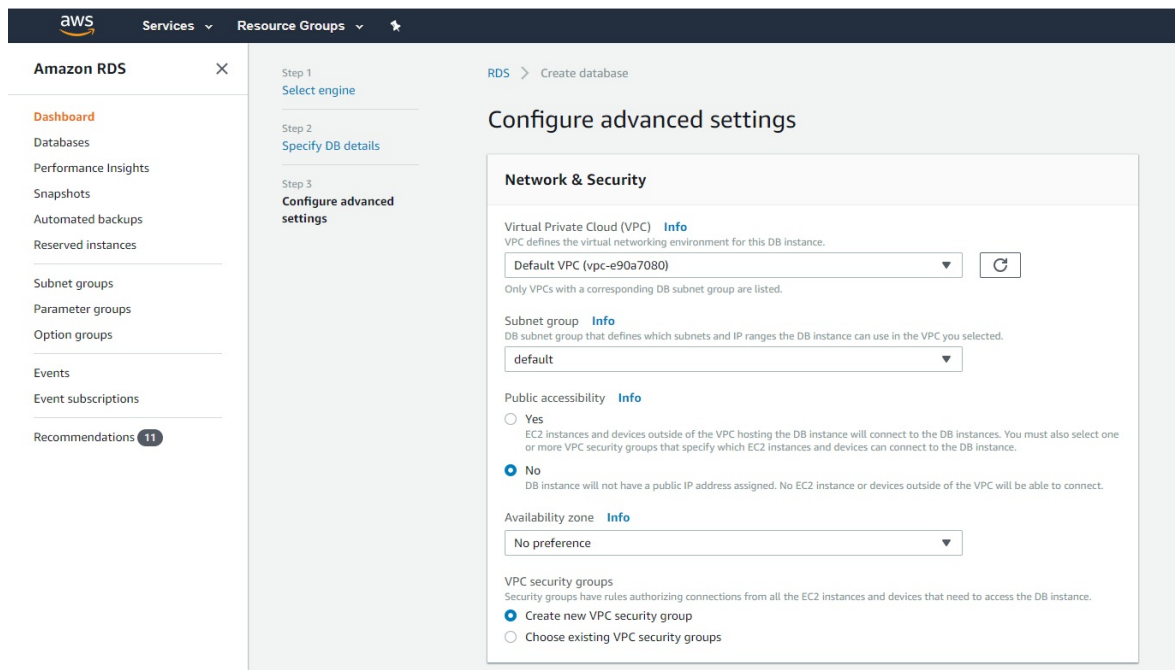


Figure 14: AWS Aurora DB

5.1.3.3.4.3.1 Network and Security

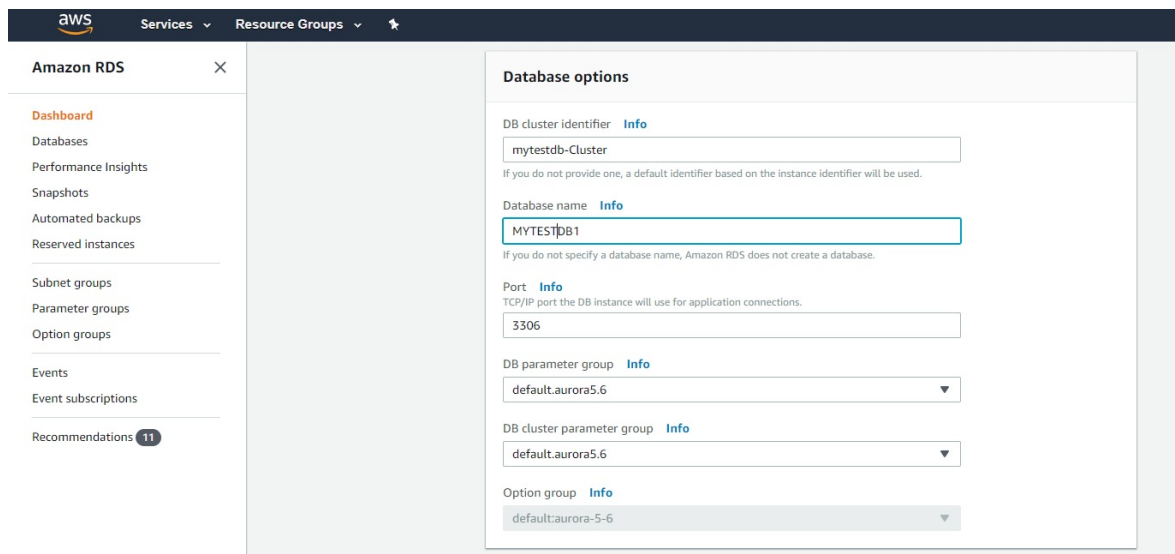
Virtual Private Cloud (VPC): This attribute defines the virtual data center under which we want to create the database. Typically an organisation will create its own VPC under which it will deploy its database and application servers. In this example we will proceed with the default VPC from AWS.

Subnet Groups: Subnets define the IP ranges to be used the DB. There can be broadly 2 types of subnets that can be created - private and public. Private subnets cannot be accessed from outside the organization network. Public subnets on the other hand have internet connectivity and can be accessed from out side the organization network.

Availaiblity zone: Having the servers across multiple availability zones with a region help prevent against localized issues and failures. AWS allows to select a preference for an availabiblity zone within the region. We dont have a specific preference for this example.

5.1.3.3.4.4 Step 6: Configure Advance Settings - Database Options

In this section we can specify the database cluster, database name, port and parameter groups (see [Figure 15](#)).



The screenshot displays the AWS Management Console interface for configuring an Amazon RDS instance. The left sidebar shows the navigation menu with 'Amazon RDS' selected. The main content area is titled 'Database options' and contains the following configuration fields:

- DB cluster identifier:** mytestdb-Cluster
- Database name:** MYTESTDB1
- Port:** 3306
- DB parameter group:** default.aurora5.6
- DB cluster parameter group:** default.aurora5.6
- Option group:** default:aurora-5-6

Figure 15: AWS Aurora DB

5.1.3.3.4.5 Step 7: Configure Advance Settings - Encryption and Failover

Aurora database has options to configure encryption and failover policies to keep the data safe and highly available. These options are explained in the next few steps(see [Figure 16](#)).

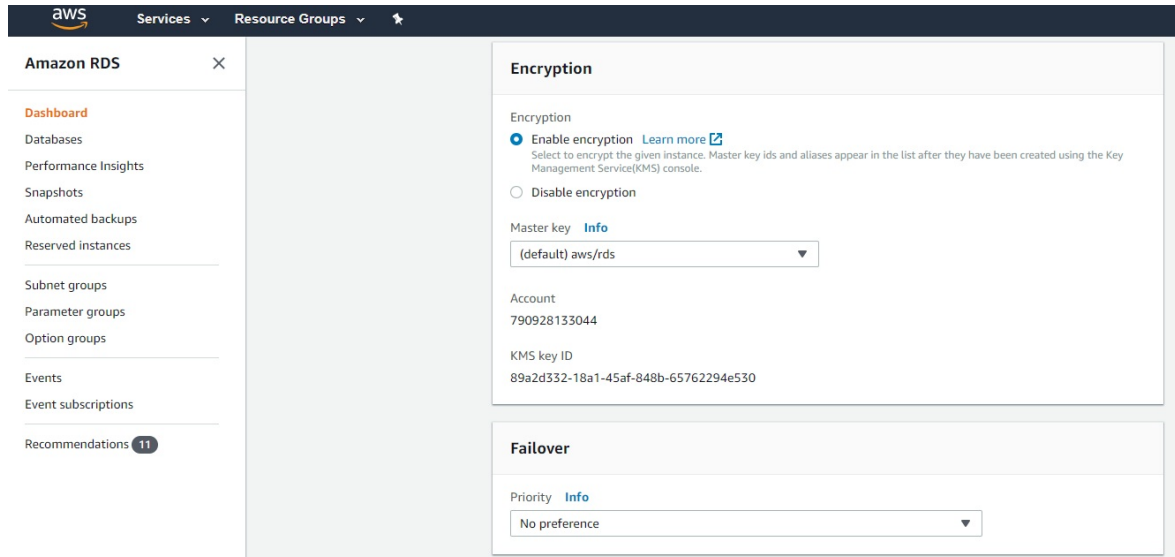


Figure 16: AWS Aurora DB

5.1.3.3.4.5.1 Encryption

Encryption on the database can be enabled only at the time of DB creation. Once enabled the database, associated replicas and snapshots are all encrypted. One can either use the default KMS encryption keys provided by AWS or use their own encryption keys.

5.1.3.3.4.5.2 Failover

This defines the the failure over priority order with which a read replica can be promoted as the primary node incase of a failure on the current writer.

5.1.3.3.4.6 Step 8: Configure Advance Settings - Backtrack and Monitoring

We can also configure settings for Backtrack and Monitoring of the database. These options are explained in the next few steps (see [Figure 17](#)).

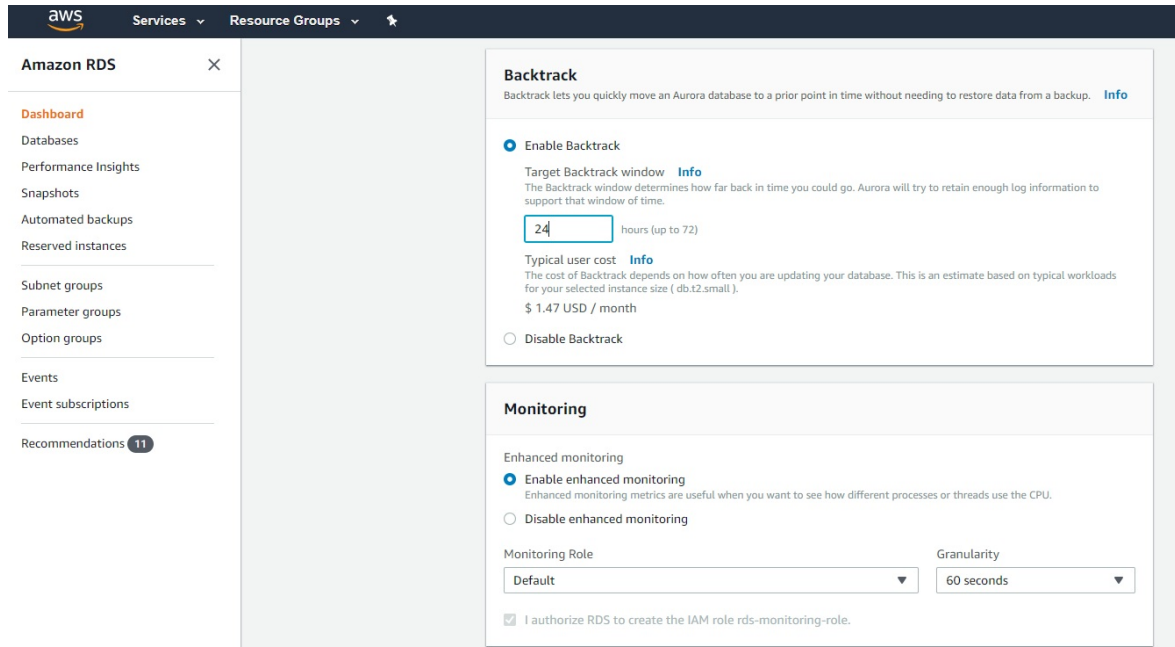


Figure 17: AWS Aurora DB

5.1.3.3.4.6.1 Backtrack

Enabling this feature allows us to define a backtrack window in hours up to which one can do a point in time recovery in case of any failure. Number of hours of backtrack define the additional charges that we levied for the database.

5.1.3.3.4.6.2 Enhanced Monitoring

Enabling enhanced monitoring allows to define the granularity and time frequency at which AWS will monitoring the database for different metrics like performance, active sessions, CPU utilization, storage used etc for reporting.

5.1.3.3.4.7 Step 9: Configure Advance Settings - Log Exports and Maintenance

Options for exporting logs and maintenance of the database can be configure as well. These options are explained in the next few steps (see [Figure 18](#)).

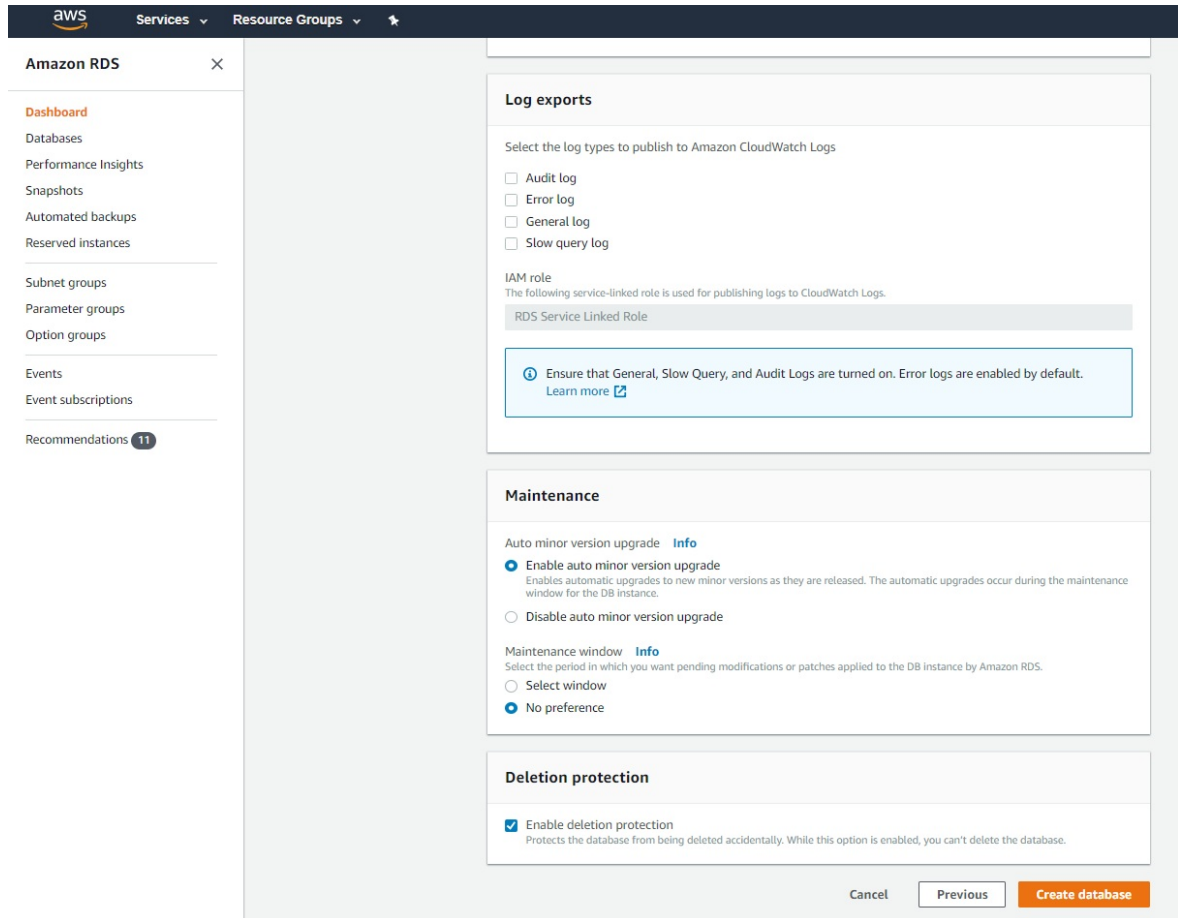


Figure 18: AWS Aurora DB

5.1.3.3.4.7.1 Log Exports

User can define which types of logs (Audit, Error, General, Slow Query) AWS will export. The selected logs options are exported to AWS S3 buckets and can be retrieved at a later point of time for analysis.

5.1.3.3.4.7.2 Maintenance

Maintenance section gives options to enable and disable auto minor version upgrades to the database. No need to worry about updating minor software packages. One can also define a preferred window for upgrades.

5.1.3.3.4.7.3 Deletion protection

Selecting this option helps prevent against accidental deletion. This option needs to be disabled first before the database can be deleted.

5.1.3.3.4.8 Step 10: Submit Advance Settings Page

Click on submit once the advance config page is completed. To view the instance creation status, click on View DB Instance button (see [Figure 19](#)).

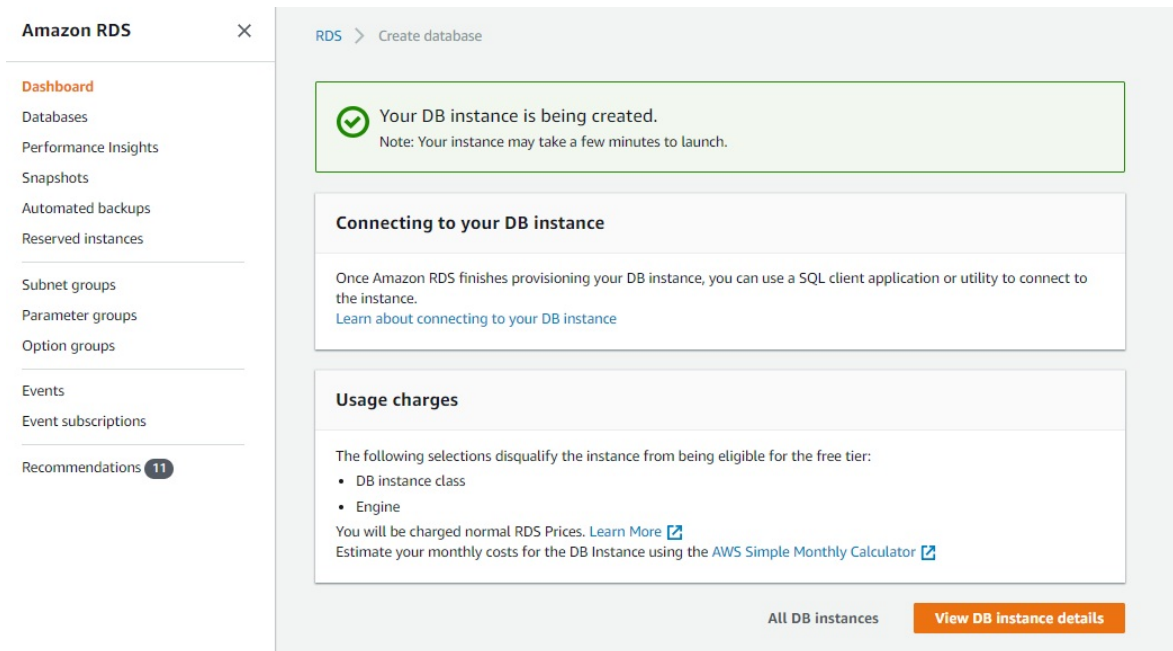


Figure 19: AWS Aurora DB

5.1.3.3.5 Step 11: Wait for DB creation to complete

Search using the db identifier entered earlier. The writer and reader db nodes will be created under a single cluster. It takes around 20-25 minutes for the servers to be available for use (see [Figure 20](#)).

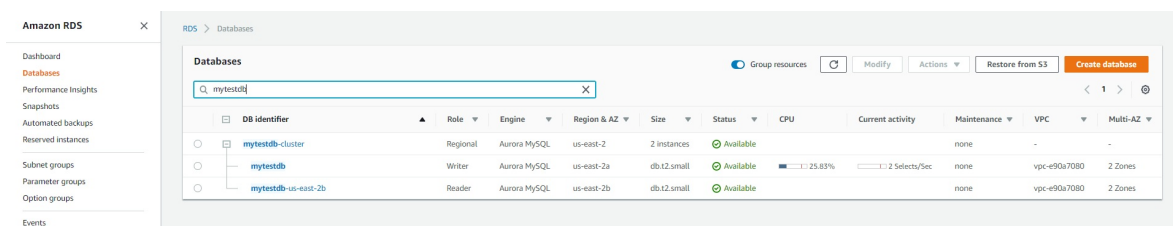


Figure 20: AWS Aurora DB

5.1.3.4 How to retrieve Aurora database connection details

The provisioned Aurora Database will give a writer and reader node. Writer node can be used to insert, update, or delete data into the database. Reader

node(s) replicate the data from the writer with a very small latency (in microseconds) and can be used to serve as the standalone node to run parallel queries for analysis without putting any additional load on the primary writer node.

The endpoint url, port needed to connect to the database can be retrieved by checking the Connectivity and Security tab (see [Figure 21](#)).

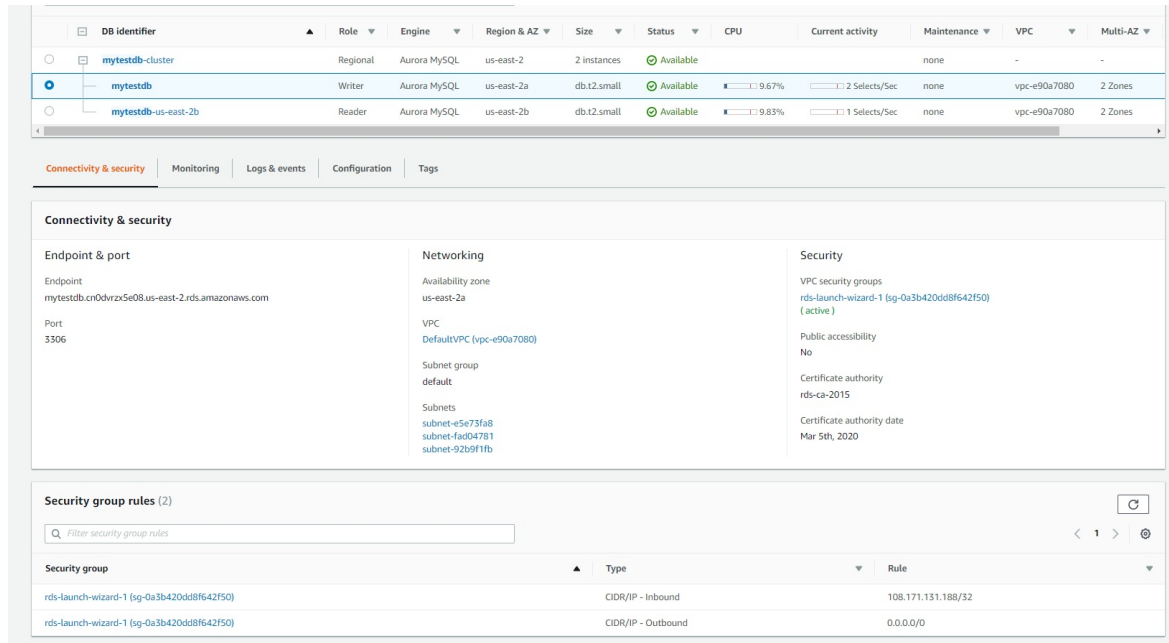


Figure 21: AWS Aurora DB

5.1.3.5 Other details available in Aurora DB Console

5.1.3.5.1 Monitoring

The Monitoring tab provides different visualization based metrics like CPU Utilization, Active DB Connections, Network Throughput etc that help in the tracking database health and usage.

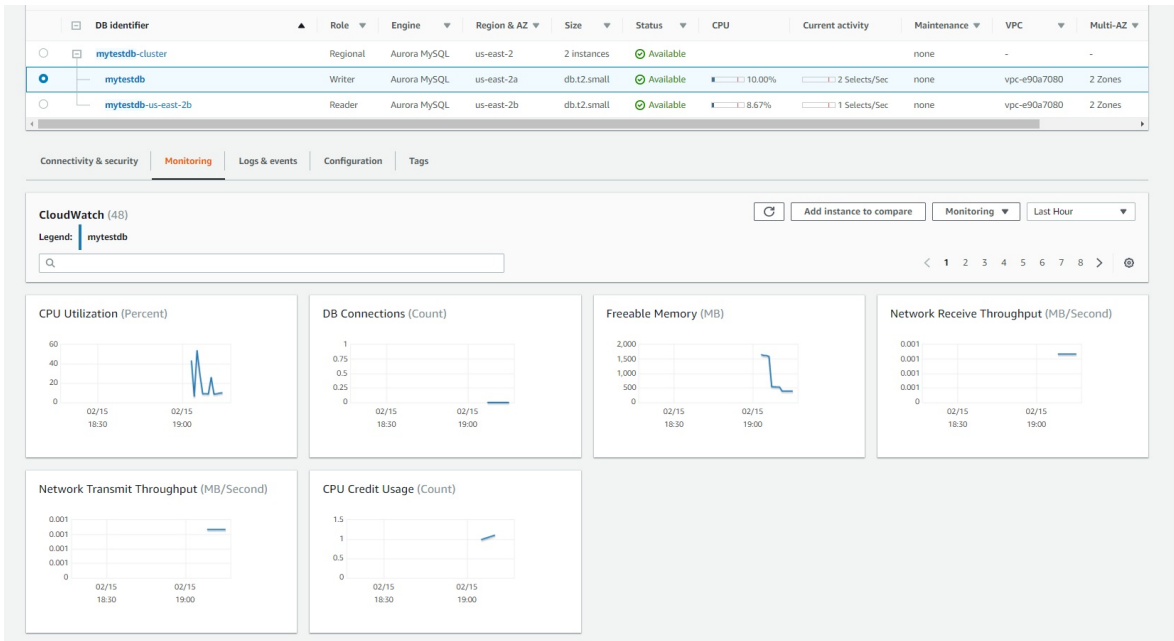


Figure 22: AWS Aurora DB

5.1.3.5.2 Logs and Events

The Logs and Events tab provides logs and event details for errors, slow running queries, long waits which can be used for debugging.

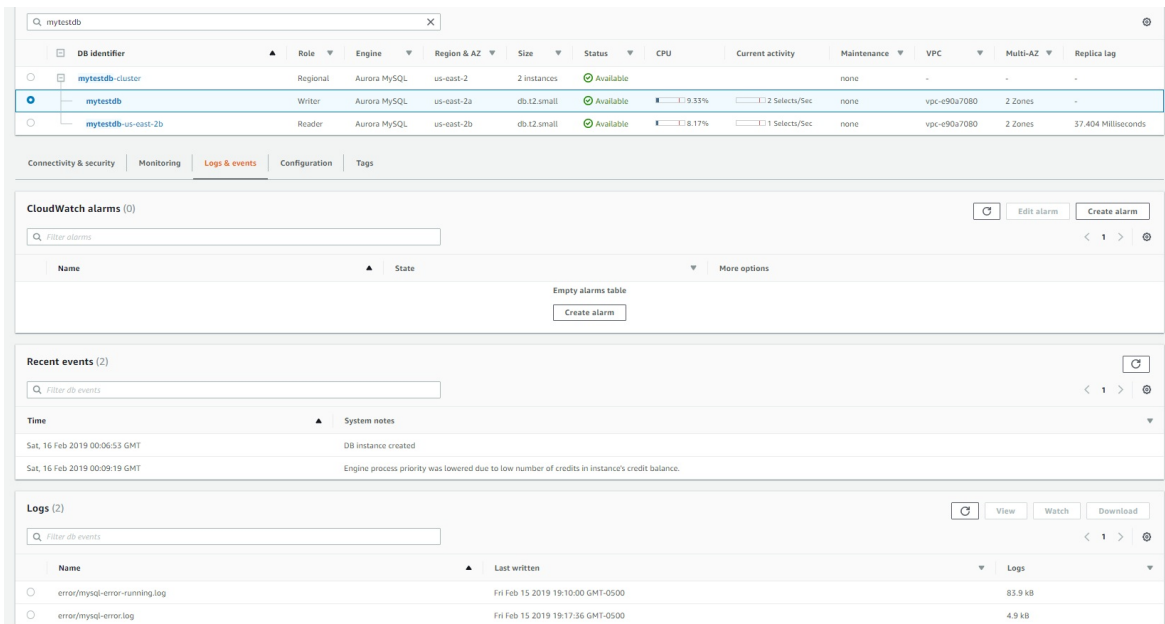


Figure 23: AWS Aurora DB

5.1.3.5.3 Configuration

The Configuration tab lists all different DB attribute values for quick reference.

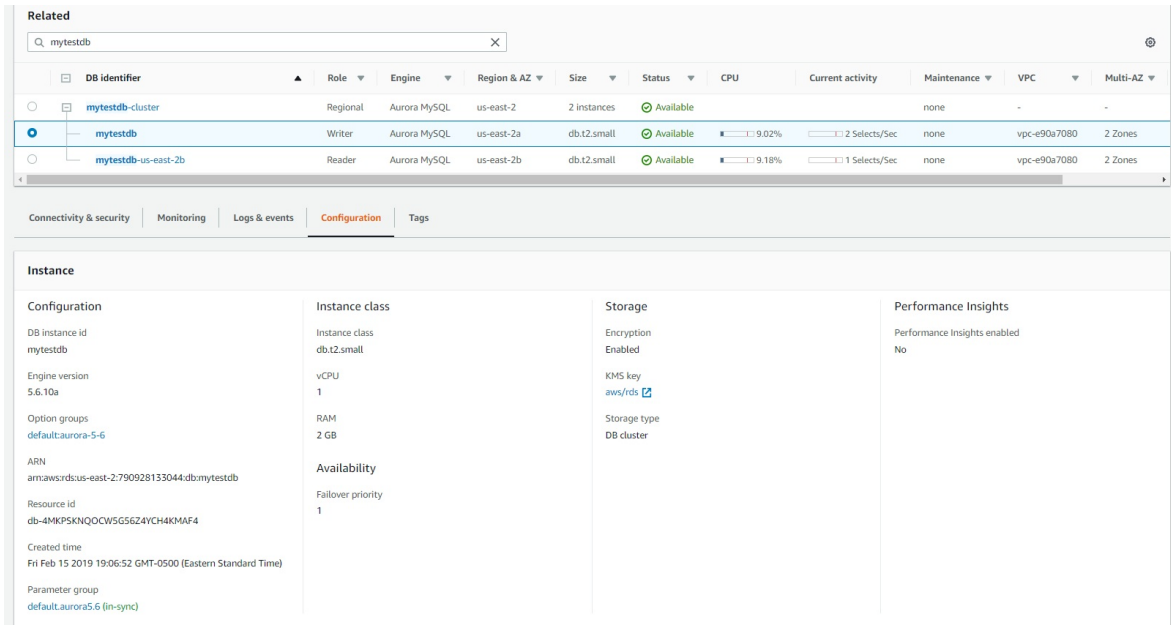


Figure 24: AWS Aurora DB

5.1.3.5.4 Tag

The Tags tab lists all tags associated with the database. Tags are often used to apply labels as key and value pairs and allow to organize the resources together logically for cost and tracking.

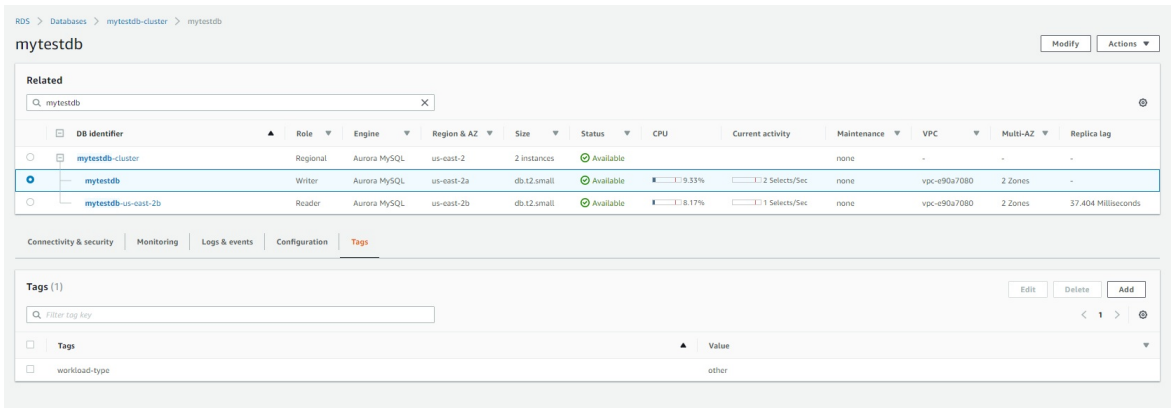


Figure 25: AWS Aurora DB

5.1.3.6 Update Aurora DB

Next we will discuss the different actions that can be taken on the Aurora

database.

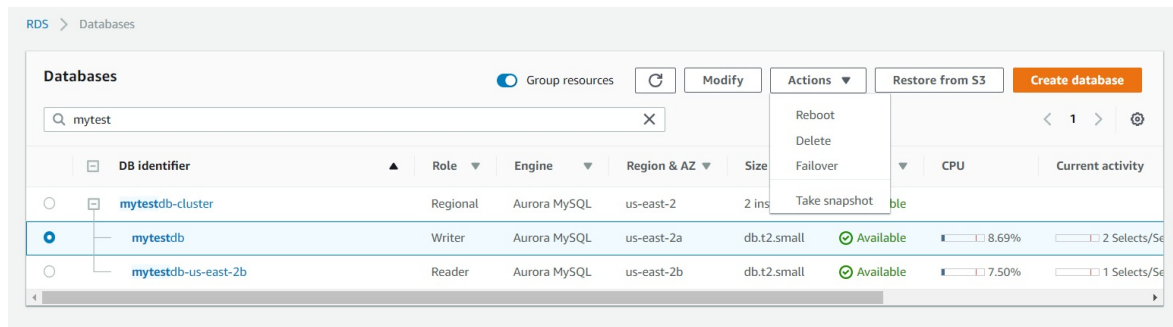


Figure 26: AWS Aurora DB

5.1.3.6.1 Modify

A database instance can be modified by clicking the `Modify` button on the top right corner.

This action allows to update the following list of attributes of Aurora DB:

- DB Instance Class
- DB Master Password
- Security Groups and Certificates
- Failover priority
- DB Port
- Backup retention period and backup window
- Backtrack Window
- Monitoring Options
- Log Export Options
- Maintenance Window

5.1.3.6.2 Reboot

This action will kill all active sessions on the database and restart the servers. All uncommitted transactions will be lost

5.1.3.6.3 Delete

This action will delete the database. One can optionally take a snapshot of the data as a backup on AWS S3.

5.1.3.6.4 Failover

This action will make the next reader node as per the failover priority as the primary writer node.

5.1.3.6.5 Take Snapshot

This action allows to take a back up of the database and store to S3 use for restore for point in time recovery.

5.1.3.7 Example

Once the database is set up and running, next step is to be able to connect and execute database queries.

Following is an example describing how to connect to AuroraDB using python.

Install MySQL driver using pip

```
$ pip install mysql-connector
```

Once the python package is installed, the below python script can be used to connect to the database, create a table and list the tables in the database.

```
import mysql.connector

mysqlHostName = '<Database server>'
mysqlUser = '<Database User>'
mysqlPassword = '<Database User Password>'
mysqlDatabase = '<Database Name>'
mysqlPort = <Database Port>

mydb = mysql.connector.connect(
    host=mysqlHostName,
    user=mysqlUser,
    passwd=mysqlPassword,
    database=mysqlDatabase,
    port=mysqlPort
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE my_user_table (name VARCHAR(255), address VARCHAR(255))")

mycursor.execute("SHOW TABLES")

for x in mycursor:
    print(x)
```

5.1.3.8 Exercises

AuroraDB.1:

Follow the steps defined in the sections above to create an AuroraDB instance with a read replica in a different availability zone. Once the main writer instance and the read replica are available, check if you can connect to both writer and reader nodes using the respective end-points.

AuroraDB.2:

Once you are able to connect to both writer and reader nodes using the end-points, create a table in the writer node and insert some data into it. Now connect to the reader node and check if you can access the same table and read the data in it.

AuroraDB.3:

Now stop the writer node on the AuroraDB. Wait for sometime and validate that the reader node is promoted as the writer node. Once the promotion is complete, connect to the new writer node and confirm if you are able to write data into the database tables.

5.1.3.9 References

- <https://aws.amazon.com/rds/aurora/> [12]

5.2 AI

5.2.1 Artificial Intelligence Service with REST

5.2.1.1 AI and ML offerings by Cloud service providers

All major cloud service providers offer a suite of AI and ML products and services. A few of the notable services include, but are not limited to:

- [Amazon ML](#) [13]
- [Google Cloud AI](#) [14]

- [IBM Watson](#) [15]
- [Microsoft Azure ML Studio](#) [16]

A few Cloud service providers expose REST based APIs to the users and one such API is [Google Cloud Vision API](#) [17] which we will discuss in the next section and illustrate the usage with an example.

5.2.1.2 Image Analysis using Google Cloud Vision API

Google Cloud Vision API offers a powerful image analysis API and it enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy-to-use REST API [17]. The API classifies the images and can be used to detect labels, logos, faces, landmarks and text within the images. The API uses JSON for both requests and responses.

Images can be provided to the API in 3 different ways [18]:

- As a base64-encoded image string. If the image is stored locally, it can be converted into a string and pass it as the value of `image.content`
- As a Google Cloud Storage URI. Pass the full URI as the value of `image.source.imageUri`
- As a publicly-accessible HTTP or HTTPS URL. Pass the URL as the value of `image.source.imageUri`

In this section, we showcase how to use Google Cloud Vision API for label detection in an image using a REST service.

Following are the pre-requisites before we can start using the API:

- Sign in to google account and create a Google Cloud Platform Project as shown in [Figure 27](#). Make sure billing is enabled.

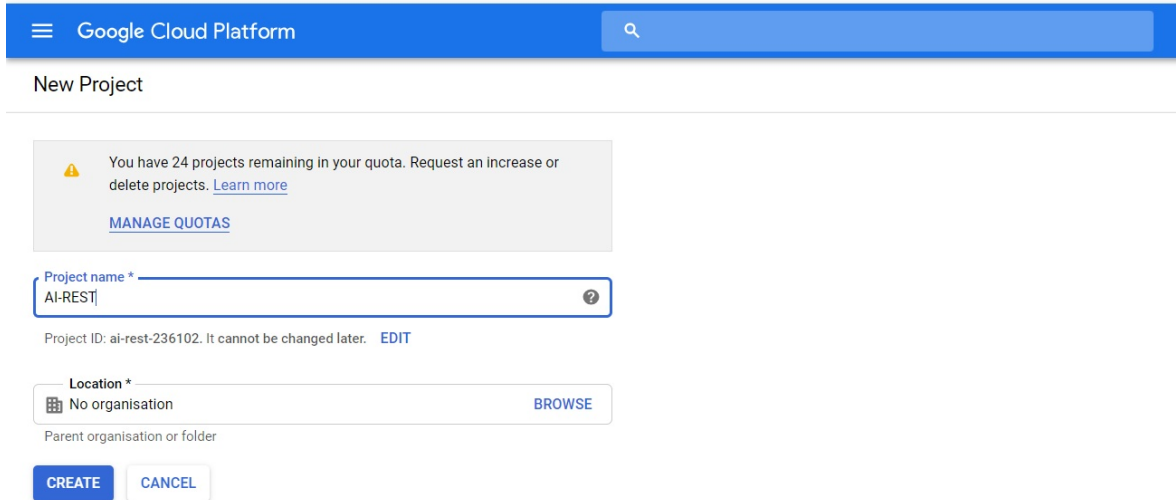


Figure 27: GCP-Project

- Enable Cloud Vision API as shown in [Figure 28](#):

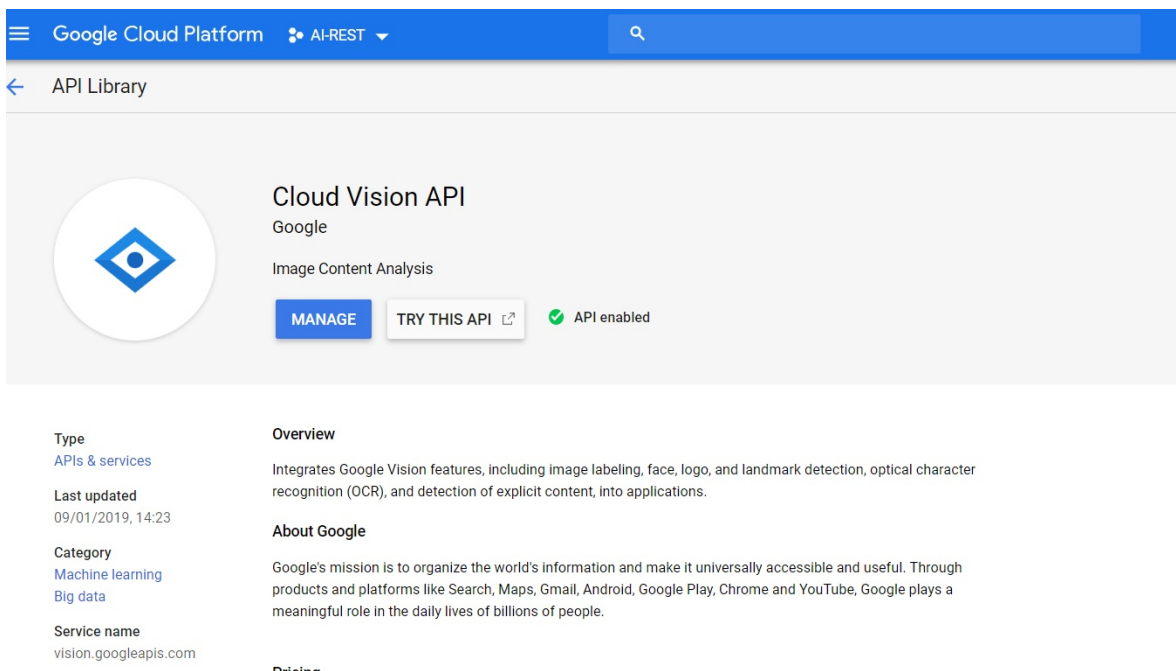


Figure 28: GCV-API

- In the GCP console, create service account key as shown in [Figure 29](#). Save the JSON file that contains the key.

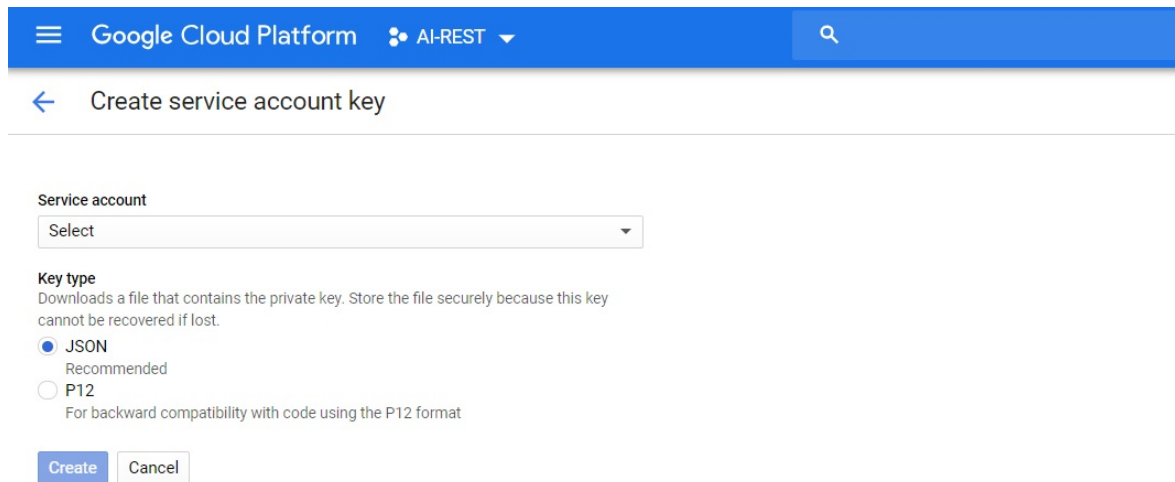


Figure 29: GCV-KEY

- Set the environment variable `GOOGLE_APPLICATION_CREDENTIALS` to the path of the JSON key file downloaded.
- Install the client library as follows:

```
$ pip install --upgrade google-cloud-vision
```

Now, we create a python module `gcv.py` for detecting labels in an image importing `google.cloud.vision` library. Image will be read from a local library and the name of the image will be passed as a parameter in the OpenAPI REST service. The function will return a simple list of all the labels detected by the API.

Note: The environment variable `GOOGLE_APPLICATION_CREDENTIALS` can also be set in the program as shown in the following code:

```
import io
import os
from flask import jsonify
# Imports the Google Cloud client library
from google.cloud import vision
from google.cloud.vision import types

def get_labels(image_name):
    current_path = os.getcwd()
    relative_path = current_path + '<name_of_key_file>.json'
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = relative_path

    # Instantiates a client
    client = vision.ImageAnnotatorClient()

    # The name of the image file to annotate
    file_name = os.path.join(current_path, image_name)

    # Loads the image into memory
```

```

with io.open(file_name, 'rb') as image_file:
    content = image_file.read()

image = types.Image(content=content)

# Performs label detection on the image file
response = client.label_detection(image=image)
labels = response.label_annotations

label_dict = {}
label_list = []
for label in labels:
    label_list.append(label.description)
label_dict['Labels'] = label_list
return jsonify(label_dict)

```

Next, we create an OpenAPI specification which will be read to invoke the API. Here we give the input image name as an inline parameter. The OpenAPI yml file `gcv.yaml` is as follows:

```

swagger: "2.0"
info:
  version: "0.0.1"
  title: "cpuinfo"
  description: "Image analysis service using swagger-2.0 specification"
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "Google Cloud Vision REST Service"
  license:
    name: "Apache"
host: "localhost:8080"
basePath: "/cloudmesh/ai"
schemes:
  - "http"
consumes:
  - "image/jpeg"
produces:
  - "application/json"
paths:
  /gcv/{image_name}:
    get:
      tags:
        - GCV
      operationId: gcv.get_labels
      description: "Returns labels detected in the image"
      parameters:
        - in: path
          name: image_name
          description: "Provide the image name in path"
          required: true
          type: string
      produces:
        - "application/json"
      responses:
        "200":
          description: "label detection"
          schema:
            $ref: "#/definitions/GCV"
definitions:
  GCV:
    type: "object"
    required:
      - "label"
    properties:
      label:
        type: "string"

```

Finally, we create a module (`server.py`) to use connexion service to read the above created OpenAPI specification (`gcv.yaml`) and dynamically call the method to be implemented on the server side.

```
"""
Main module of the server file
"""
from flask import jsonify
import connexion

# Create the application instance
app = connexion.App(__name__, specification_dir="./")

# Read the yaml file to configure the endpoints
app.add_api("gcv.yaml")

# create a URL route in our application for "/"
@app.route("/")
def home():
    msg = {"msg": "AI service with REST"}
    return jsonify(msg)

if __name__ == "__main__":
    app.run(port=8080, debug=True)
```

To implement the REST service, run the following on the terminal.

```
$ python server.py
```

Once the connection is established, run the following CURL command on the terminal

```
$ curl http://localhost:8080/cloudmesh/ai/gcv/mp1.jpg
```

Input image `mp1.jpg` is shown in [Figure 30](#).



Figure 30: GCV-Test Image

Output response from the REST API.

```
{
  "Labels": [
    "Mountainous landforms",
    "Mountain",
    "Landmark",
    "Highland",
    "Hill station",
    "Historic site",
    "Mountain range",
    "Wonders of the world",
    "Ruins",
    "Ancient history"
  ]
}
```

5.2.1.3 Naive Bayes Algorithm for Text classification

Naive Bayes is a simple yet powerful classification machine learning algorithm. In this section we demonstrate the implementation of Naive Bayes algorithm on text documents in a RESTful service to classify a review as positive or negative.

Example setup: In this example we will consider a text document containing reviews of a restaurant. Data is split into two datasets - training dataset and test dataset. For the purposes of illustrating the example, the datasets are made available in an Azure public blob storage container and the links are mentioned in the following details:

- Training data: We will utilize a pre-processed training dataset with labels attached to each review as *positive* or *negative*. Training data can be downloaded from [here](#).
- Test data: Test dataset has been setup in such a way that first 2989 reviews are positive and rows from 2990 to 4321 are negative reviews. Test data needs to pre-processed and cleaned before the algorithm is implemented. After the test data is cleaned, we will label the test data as per the information given.

Finally, we will implement Multinomial Naive Bayes classifier algorithm and calculate the accuracy of the test prediction. Test data can be downloaded from [here](#).

To implement machine learning algorithm on text documents we will use scikit-learn feature extraction modules. Please refer to related documentation in the

following scikit-learn link - [Feature Extraction](#) [19].

For the current example we will use the following specific modules:

```
sklearn.feature_extraction.text.CountVectorizer  
sklearn.feature_extraction.text.TfidfTransformer
```

Solution will be implemented in following steps:

- **Step 1:** Define a function to download data from Azure cloud storage.
- **Step 2:** Define a function to pre-process *Test* dataset.
- **Step 3:** Define a function to implement Naive Bayes algorithm.
- **Step 4:** Define an OpenAPI specification in a YAML file. The specification will have 3 endpoints for each of the previous steps:
 - Download training and test datasets.
 - Pre-process Test data with parameter.
 - Build Naive Bayes classification model and return test accuracy.
- **Step 5:** Create a module to use the connexion service and read in the OpenAPI specification from the yaml file.

Now we discuss in details, each of the steps mentioned earlier.

Pre-requisites:

Following libraries need to be imported for the current example:

```
from cloudmesh.common.util import path_expand  
import os  
import re  
from sklearn.naive_bayes import MultinomialNB  
from sklearn.feature_extraction.text import CountVectorizer  
from sklearn.feature_extraction.text import TfidfTransformer  
import numpy as np  
from flask import jsonify  
import connexion
```

Step 1: Define a function to download data from Azure cloud storage

Note: This step downloads data from Azure cloud Blob storage. Prior to replicating this step, datasets need to be uploaded to your respective Azure storage account and pertaining credentials are to be used in the program. Optionally, this step can be skipped if datasets are downloaded from the direct links provided previously (under Example setup) and get started from the next step.

As mentioned previously, training and test datasets are uploaded to Azure blob storage which can be downloaded using the Azure blob storage client libraries. Credentials and the container name will be read from a yaml file via the `Cloudmesh_community Config()` library.

```
from cloudmesh.common.util import path_expand
from cloudmesh.management.configuration.config import Config
from azure.storage.blob import BlockBlobService
import os, uuid, sys
from flask import jsonify

def download_data():
    print("enter")
    config = Config()
    block_blob_service = BlockBlobService(
        account_name=config['cloudmesh.storage.azure-2.credentials.account_name'],
        account_key=config['cloudmesh.storage.azure-2.credentials.account_key'])

    container_name=config['cloudmesh.storage.azure-2.credentials.container']

    blob_gen = block_blob_service.list_blobs(container_name)
    blob_list = []
    for blob in blob_gen:
        blob_list.append(blob.name)
        print("\t Blob name: " + blob.name)

    for i in range(len(blob_list)):
        filename = blob_list[i]
        local_path = path_expand("~/")
        full_path_to_file = os.path.join(local_path, filename)
        print("\nDownloading blob to " + full_path_to_file)
        block_blob_service.get_blob_to_path(container_name, filename, full_path_to_file)

    return 'Datasets downloaded'
```

Step 2: Define a function to pre-process *Test* dataset

Pre-processing test data involves following tasks:

- Cleaning the text data i.e. remove unwanted characters, convert all text to lower case, delete any extra spaces and finally join all the words together into sentences.
- Label the rows as per the information provided i.e. label first 2989 rows as *positive* and rest as *negative*. We will take 2990 as a input parameter when we define the endpoint.

```
from cloudmesh.common.util import path_expand
import os
import re

def process_testfile(linenum):
    test_fp = os.path.join(path_expand("~/"), 'testSet.txt')
    processed_test = os.path.join(path_expand("~/"), 'processedTest.csv')
    test_file = open(test_fp)
    lines = test_file.readlines()
    write_test = open(processed_test, "w")

    # Label 1st 'linenum' lines as positive
    file_clean(lines[:linenum], "Positive", write_test)

    # Label lines after 'linenum' count as negative
```

```

file_clean(lines[linenum:], "Negative", write_test)

test_file.close()
write_test.close()

# Internal function for File Cleanup
def file_clean(infile, label, writeFile):
    badChar = "[,!?.#@=\n]"

    for line in infile:
        line = line.lower().replace("\t", " ")
        line = re.sub(badChar, "", line)
        arr = line.split(" ")
        words = " ".join(word for word in arr)
        toWrite = label+","+words
        writeFile.write(toWrite)
        writeFile.write("\n")

```

Step 3: Define a function to implement Naive Bayes algorithm

After the Test dataset has been cleaned and labelled, we now run the Multinomial Naive Bayes algorithm on training data and use the model to classify test data.

As mentioned earlier, we will use scikit-learn feature selection modules CountVectorizer and TfidfTransformer to transform the text into numerical feature vectors and downscale weights for words that occur in many data points but are less informative (like ‘a’, ‘is’, ‘the’, etc.)

```

from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import numpy as np

def naivebayes():

    processed_train = os.path.join(path_expand("~/"), 'processedTrain.csv')
    processed_test = os.path.join(path_expand("~/"), 'processedTest.csv')

    # get the data and label for training and test data
    data_train, label_train = get_data_label(processed_train)
    data_test, label_test = get_data_label(processed_test)

    count_vect = CountVectorizer()
    X_train_counts = count_vect.fit_transform(data_train)
    tfidf_transformer = TfidfTransformer()
    X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

    model = MultinomialNB(fit_prior=True)
    model.fit(X_train_tfidf, label_train)
    X_new_counts = count_vect.transform(data_test)
    X_new_tfidf = tfidf_transformer.transform(X_new_counts)

    predLabel = model.predict(X_new_tfidf)

    nb_result = {}
    nb_result["Test Accuracy"] = round(np.mean(predLabel == label_test) * 100, 2)
    return jsonify(nb_result)

# Internal Function to fetch Data and labels
def get_data_label(inp_file):
    file = open(inp_file) # read the processed file
    label = []
    data = []

```

```

for line in file:
    arr = line.replace("\n", "").split(",") # split with comma
    label.append(arr[0]) # first element is class label
    data.append(arr[1].replace("\n", "")) # second element is SMS
return data, label

```

Step 4: Define an OpenAPI specification in a YAML file

Now we define an OpenAPI specification in yaml format to create 2 different endpoints for functions defined in the revios 3 steps.

- Functions defined in step-1, step-2 and step-3 need to be part of a module named `ai.py`
- `operationId` in the specification will correspond to the names of the functions defined in each of the previous steps respectively.
- The input parameter required for function in step-2 will be passed as inline parameter (`linenum`) for the endpoint to pre-process test dataset.

```

swagger: "2.0"
info:
  version: "0.0.1"
  title: "naivebayes"
  description: "A service to run a Naive Bayes ML using swagger-2.0 specification"
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "Naive Bayes ML algorithm REST Service"
  license:
    name: "Apache"
host: "localhost:8080"
basePath: "/cloudmesh/ai"
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:
  /getdata:
    get:
      tags:
        - AI
      operationId: ai.get_data
      description: "runs naive bayes alogirthm"
      produces:
        - "multipart/form-data"
        - "application/json"
      responses:
        "200":
          description: "naive bayes ml"
          schema:
            $ref: "#/definitions/AI"
  /testdata/{linenum}:
    get:
      tags:
        - AI
      operationId: ai.process_testfile
      description: "runs naive bayes alogirthm"
      parameters:
        - in: path
          name: linenum
          description: "Provide the line number in path"
          required: true
          type: integer
      produces:
        - "multipart/form-data"
        - "application/json"
      responses:

```

```

    "200":
      description: "naive bayes ml"
      schema:
        $ref: "#/definitions/AI"
  /nb:
    post:
      tags:
        - AI
      operationId: ai.naivebayes
      description: "uploads output data to azure"
      consumes:
        - "multipart/form-data"
      produces:
        - "application/json"
      responses:
        "200":
          description: "naive bayes ml"
          schema:
            $ref: "#/definitions/AI"
definitions:
  AI:
    type: string

```

Step 5: Create a module to use the connexion service to start the server

Finally, we create a module (`server.py`) to use connexion service to read the above created OpenAPI specification (`ai.yaml`) and dynamically call the methods to be implemented on the server side.

```

"""
Main module of the server file
"""
from flask import jsonify
import connexion

# Create the application instance
app = connexion.App(__name__, specification_dir="./")

# Read the yaml file to configure the endpoints
app.add_api("ai.yaml")

# create a URL route in our application for "/"
@app.route("/")
def home():
    msg = {"msg": "AI service with REST"}
    return jsonify(msg)

if __name__ == "__main__":
    app.run(port=8080, debug=True)

```

To implement the REST service, run the following on the terminal.

```
$ python server.py
```

Once the connection is established, following CURL command can be used for the 1st endpoint which will download training and test datasets to local library.

```
$ curl http://localhost:8080/cloudmesh/ai/getdata
```

Following CURL command can be used for the 2nd endpoint which will pre-process the test dataset

```
$ curl http://localhost:8080/cloudmesh/ai/testdata/2990
```

Following CURL command can be used for the 3rd endpoint which will build the model to classify the test data and finally return the accuracy

```
$ curl http://localhost:8080/cloudmesh/ai/nb
```

5.3 LINUX

5.3.1 Windows Subsystem for Linux



5.4 DEVOPS

5.4.1 Infrastructure as Code (IaC)

5.4.1.1 Learning Objectives



Learning Objectives

- Introduction to IaC
 - How IaC is related to DevOps
 - How IaC differs from Configuration Management Tools, and how is it related
 - Listing of IaC Tools
 - Further Reading
-

5.4.1.2 Introduction to IaC

IaC(Infrastructure as Code) is the ability of code to generate, maintain and destroy application infrastructure like server, storage and networking, without requiring manual changes. State of the infrastructure is maintained in files.

Cloud architectures, and containers have forced usage of IaC, as the amount of elements to manage at each layer are just too many. It is impractical to keep track with the traditional method of raising tickets and having someone do it for you. Scaling demands, elasticity during odd hours, usage-based-billing all require provisioning, managing and destroying infrastructure much more dynamically.

From the book “Amazon Web Services in Action” by Wittig [20], using a script or a declarative description has the following advantages

- *Consistent usage*
- *Dependencies are handled*
- *Replicable*
- *Customizable*
- *Testable*
- *Can figure out updated state*
- *Minimizes human failure*
- *Documentation for your infrastructure*

Sometimes IaC tools are also called Orchestration tools, but that label is not as accurate, and often misleading.

5.4.1.3 How IaC is related to DevOps

DevOps has the following key practices * Automated Infrastructure * Automated Configuration Management, including Security * Shared version control between Dev and Ops * Continuous Build - Integrate - Test - Deploy * Continuous Monitoring and Observability

The first practice - Automated Infrastructure can be fulfilled by IaC tools. By having the code for IaC and Configuration Management in the same code repository as application code ensures adhering to the practice of shared version control.

Typically, the workflow of the DevOps team includes running Configuration Management tool scripts after running IaC tools, for configurations, security, connectivity, and initializations.

5.4.1.4 How IaC tools differs from Configuration Management Tools, and how it is related

There are 4 broad categories of such tools [21], there are > * Ad hoc scripts: Any shell, Python, Perl, Lua scripts that are written > * Configuration management tools: Chef, Puppet, Ansible, SaltStack > * Server templating tools: Docker, Packer, Vagrant > * Server provisioning tools: Terraform, Heat, CloudFormation, Cloud Deployment Manager, Azure Resource Manager

Configuration Management tools make use of scripts to achieve a state. IaC tools maintain state and metadata created in the past.

However, the big difference is the state achieved by running procedural code or scripts may be different from state when it was created because * Ordering of the scripts determines the state. If the order changes, state will differ. Also, issues like waiting time required for resources to be created, modified or destroyed have to be correctly dealt with. * Version changes in procedural code are inevitable, and will lead to a different state.

Chef and Ansible are more procedural, while Terraform, CloudFormation, SaltStack, Puppet and Heat are more declarative.

IaC or declarative tools do suffer from inflexibility related to expressive scripting language.

5.4.1.5 Listing of IaC Tools

IaC tools that are cloud specific are Amazon AWS - AWS CloudFormation
Google Cloud - Cloud Deployment Manager
Microsoft Azure - Azure Resource Manager
OpenStack - Heat

Terraform is not a cloud specific tool, and is multi-vendor. It has got good support for all the clouds, however, Terraform scripts are not portable across clouds.

5.4.1.6 Advantages of IaC

IaC solves the problem of *environment drift*, that used to lead to the infamous “but it works on my machine” kind of errors that are difficult to trace. According to ???

IaC guarantees Idempotence – known/predictable end state – irrespective of starting state. Idempotency is achieved by either automatically configuring an existing target or by discarding the existing target and recreating a fresh environment.

5.4.1.7 Further Reading

Please see books and resources like the “Terraform Up and Running” [21] for more real-world advice on IaC, structuring Terraform code and good deployment practices.

A good resource for IaC is the book “Infrastructure as Code” [22].

5.5 OTHER

5.5.1 Amazon Elastic Beanstalk



Learning Objectives

- Learn about AWS Elastic Beanstalk
 - Benefits
 - Pricing
 - How to provision a Elastic Beanstalk Environment for Quick Start
 - Explore Elastic Beanstalk Features
-

Amazon Elastic Beanstalk is an PaaS offering from AWS that aims to be an one stop solution for fast deployment of scalable web-applications.

Elastic Beanstalk has been developed with an idea of allowing developers to focus on code development instead of environment set up. All that the developers need to do is to provide a working code, beanstalk takes care of

providing the platform to run it.

A Variety of Platforms are supported

- Java
- Python
- Node JS
- .NET
- PHP
- Ruby
- Go
- Docker

Features of Elastic Beanstock include

- *High Performance*: Elastic Beanstalk allows freedom to provision servers from a variety of EC2 configurations based on business to suit the compute requirements.
- *Scalability*: Easy configurable auto scaling settings are available to handle peak loads based on metrics monitoring.
- *High Availability*: Elastic Beanstalk provides easy health check based load balancing options to ensure high availability of the web application.
- *Complete source control*: The user has freedom to select the AWS resources and has full control over the infrastructure powering the application using Elastic Beanstalk management capabilities.
- *Fully Managed*: With Elastic Beanstalk, AWS manages activities like hardware provisioning, set up and configuration, software patching, database backups and performance monitoring.

5.5.1.1 Amazon Elastic Beanstalk Pricing

Amazon Elastic Beanstalk has no additional charges of its own. You pay for the resources and services provisioned by the beanstalk environment. These are typically: * EC2 instances * S3 Storage * Elastic Load Balancer * Database

5.5.1.2 How to provision a Amazon Elastic Beanstalk Environment for Quick Start

In this section we will discuss the steps to be followed to create a Highly Available, Load Balanced sample application on Elastic Beanstalk in a few quick Steps.

To be able to create a beanstalk application, the user must have set up an AWS account as a pre-requisite. An AWS account can be created using the link below

[New AWS account creation URL](#)

5.5.1.2.1 Step 1: Login to the AWS console.

Login to the AWS account using the link below.

[AWS Console URL](#).

Upon successful login, select Elastic Beanstalk from the Compute section or alternatively, you can type Beanstalk in the search bar to look up (see [Figure 31](#)).

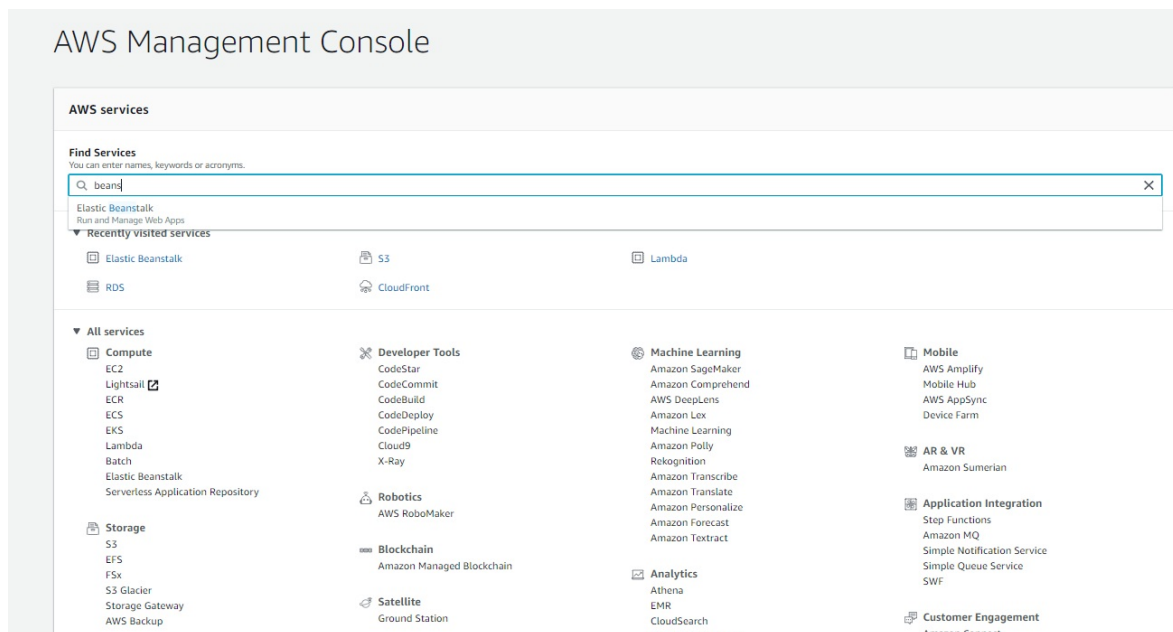


Figure 31: AWS Beanstalk

5.5.1.2.2 Step 2: Click on Create New Application

On the Beanstalk home page, one can either click on the `Create New Application` button at the top right corner or click on the `Get Started` button in the screen center (see [Figure 32](#)).

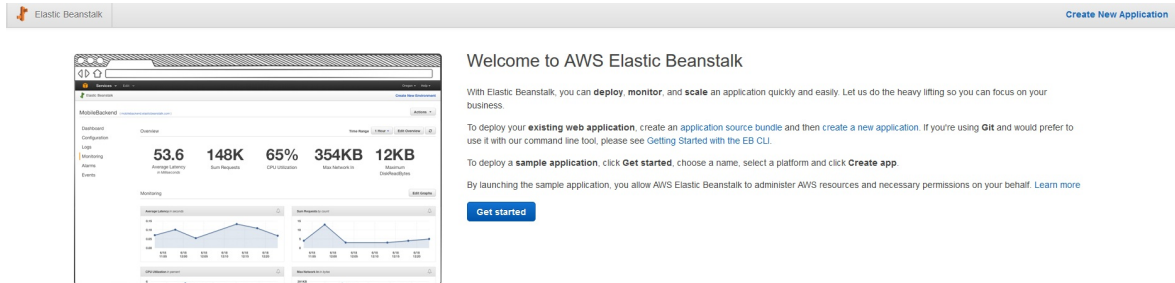


Figure 32: AWS Beanstalk

5.5.1.2.3 Step 3: Select Base Configuration

Provide a suitable name for your application (see [Figure 33](#)). In the Base Configuration section, select the type of platform you wish to provision for your code to be executed. There are several options available like Java, Python, Ruby, Go, .NET, PHP, Node JS, Docker and Tomcat (see [Figure 34](#)). For this example we will go with Tomcat.

Create a web app

Create a new application and environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)

Application information

Application name

Up to 100 Unicode characters, not including forward slash (/).

Base configuration

Platform

Choose [Configure more options](#) for more platform configuration options.

Application code

Sample application
Get started right away with sample code.

Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

ZIP or WAR

Figure 33: AWS Beanstalk

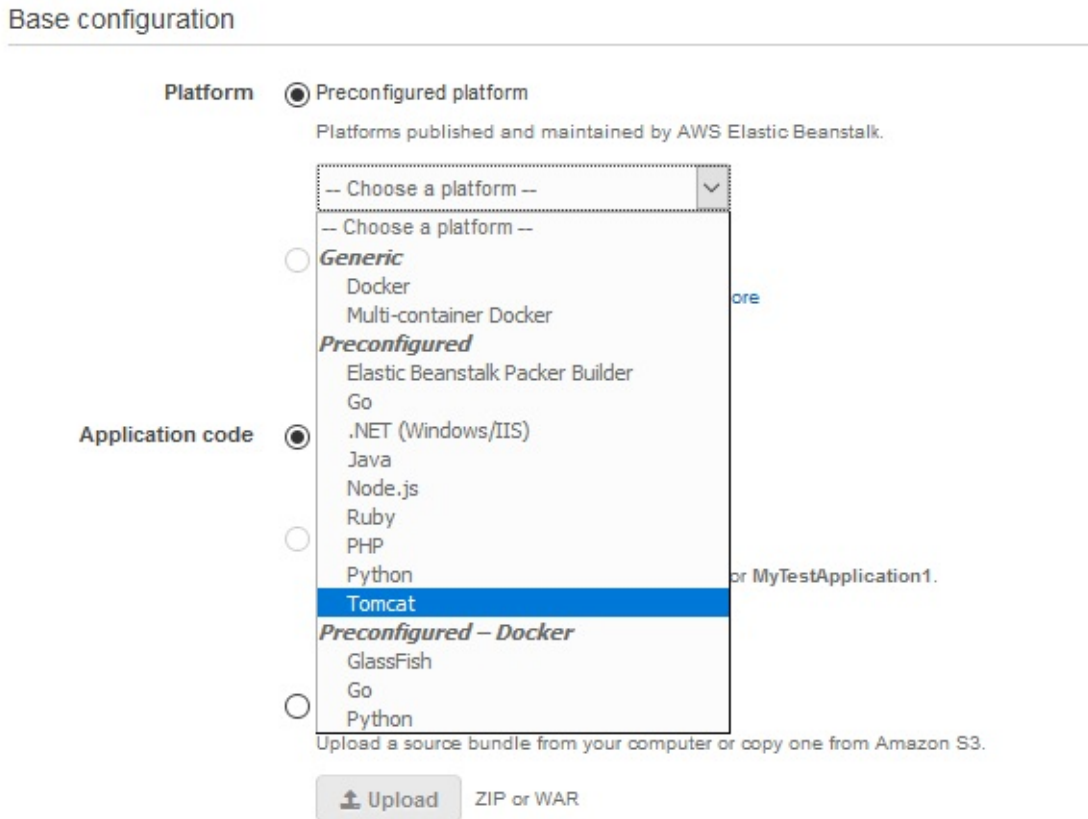


Figure 34: AWS Beanstalk

5.5.1.2.4 Step 4: Click on Configure More Options

Do not click on Create Application button in the previous step in case you wish to create your application in a particular VPC. This will be a requirement for most organizations since they would like the application to be created in their own VPCs on cloud so that they can restrict the access to the application and also allow the application to connect with other resources on their cloud like databases. If you go ahead and click on Create Application button, the application will be created in default AWS VPC. Once an application is created, the VPC on it cannot be updated.

To be able to launch the application in the VPC of your choice, click on [configure More Options](#) button.

5.5.1.2.5 Step 5: Select High Availability as Config Preset

Once you click on `Configure More Options` button, AWS will take you to Advance Configuration page. Select High Availability option as configuration present. This will allow to provision a load balancer for your application. If this option is not selected, the application is launched as low cost and no load balancer can be mapped to it (see [Figure 35](#)).

Configure Mytestapplication-env

Start from a preset that matches your use case or choose *Custom configuration* to unset recommended values and use the service's default values.

Configuration presets Low cost (*Free Tier eligible*)
 High availability
 Custom configuration

Platform Tomcat 8.5 with Java 8 running on 64bit Amazon Linux/3.1.0 [Change platform configuration](#)

Software AWS X-Ray: disabled Rotate logs: disabled (default) Log streaming: disabled (default) Environment properties: 1 JDBC_CONNECTION_STRING	Instances EC2 Instance type: t2.micro EC2 Image ID: ami-046c0289c985f5ed3 Root volume type: container default Root volume size (GB): container default Root volume IOPS: container default Security groups: none	Capacity Environment type: load balancing, auto scaling Availability Zones: Any Instances: 1-4
Load balancer Load balancer type: application Listeners: 1 Processes: 1 Rules: 1	Rolling updates and deployments Deployment policy: All at once Rolling updates: disabled Health check: enabled	Security Service role: aws-elasticbeanstalk-service-role Virtual machine key pair: -- Virtual machine instance profile: aws-elasticbeanstalk-ec2-role
Monitoring Health reporting system: Enhanced Ignore HTTP 4xx: disabled Health event log streaming: disabled	Managed Updates Managed updates: disabled	Notifications Email address: --
Network This environment is not part of a VPC.	Database Engine: -- Instance class: -- Storage (GB): -- Multi-AZ: --	Tags Tags: none

[Cancel](#) [Previous](#) [Create app](#)

Figure 35: AWS Beanstalk

5.5.1.2.6 Step 6: Map VPC

Click on the Network section on the Advance Configuration page (see [Figure 35](#)). This will open the another page which will allow to select the VPC under which we want the application to be launched. This step cannot be reversed so pay special attention to the VPC that is selected here. For this example, we will continue with the default VPC (see [Figure 36](#)).

Modify network

Virtual private cloud (VPC)

Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console. [Learn more](#)

VPC 

[Create custom VPC](#)

Load balancer settings

Assign your load balancer to a subnet in each Availability Zone (AZ) in which your application runs. For a publicly accessible application, set **Visibility** to **Public** and choose public subnets.

Visibility

Make your load balancer internal if your application serves requests only from connected VPCs. Public load balancers serve requests from the Internet.

Load balancer subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-435b91f1	172.31.32.0/20	
<input checked="" type="checkbox"/>	us-east-1b	subnet-92e547f5	172.31.0.0/20	
<input type="checkbox"/>	us-east-1c	subnet-6e03af40	172.31.80.0/20	
<input type="checkbox"/>	us-east-1d	subnet-0f80f145	172.31.16.0/20	
<input type="checkbox"/>	us-east-1e	subnet-3e6c9d00	172.31.64.0/20	
<input type="checkbox"/>	us-east-1f	subnet-ef224fe0	172.31.48.0/20	

Instance settings

Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances.

Public IP address Assign a public IP address to the Amazon EC2 instances in your environment.

Instance subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-435b91f1	172.31.32.0/20	
<input checked="" type="checkbox"/>	us-east-1b	subnet-92e547f5	172.31.0.0/20	
<input type="checkbox"/>	us-east-1c	subnet-6e03af40	172.31.80.0/20	
<input type="checkbox"/>	us-east-1d	subnet-0f80f145	172.31.16.0/20	
<input type="checkbox"/>	us-east-1e	subnet-3e6c9d00	172.31.64.0/20	
<input type="checkbox"/>	us-east-1f	subnet-ef224fe0	172.31.48.0/20	

Cancel

Save

Figure 36: AWS Beanstalk

Other options that can be configured in this step are explained next.

5.5.1.2.6.1 Load Balancer settings

Select visibility as public in case you want your application to be accessible from other VPCs or over internet.

User can select the subnet(s) to be used for the load balancer across different availability zones.

These settings can be modified after the application launch as well.

5.5.1.2.6.2 Instance settings

You can assign public IP address to your EC2 instances from this section. However this action is discouraged and the application should be accessible only through the load balancers.


User can select the subnet(s) to be used for the application servers across different availability zones.

These settings can be modified after the application launch as well.

Modify network

Virtual private cloud (VPC)

Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console. [Learn more](#)

VPC 

[Create custom VPC](#)

Load balancer settings

Assign your load balancer to a subnet in each Availability Zone (AZ) in which your application runs. For a publicly accessible application, set **Visibility** to **Public** and choose public subnets.

Visibility

Make your load balancer internal if your application serves requests only from connected VPCs. Public load balancers serve requests from the Internet.

Load balancer subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-435b91f1	172.31.32.0/20	
<input checked="" type="checkbox"/>	us-east-1b	subnet-92e547f5	172.31.0.0/20	
<input type="checkbox"/>	us-east-1c	subnet-6e03af40	172.31.80.0/20	
<input type="checkbox"/>	us-east-1d	subnet-0f80f145	172.31.16.0/20	
<input type="checkbox"/>	us-east-1e	subnet-3e6c9d00	172.31.64.0/20	
<input type="checkbox"/>	us-east-1f	subnet-ef224fe0	172.31.48.0/20	

Instance settings

Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances.

Public IP address Assign a public IP address to the Amazon EC2 instances in your environment.

Instance subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-435b91f1	172.31.32.0/20	
<input checked="" type="checkbox"/>	us-east-1b	subnet-92e547f5	172.31.0.0/20	
<input type="checkbox"/>	us-east-1c	subnet-6e03af40	172.31.80.0/20	
<input type="checkbox"/>	us-east-1d	subnet-0f80f145	172.31.16.0/20	
<input type="checkbox"/>	us-east-1e	subnet-3e6c9d00	172.31.64.0/20	
<input type="checkbox"/>	us-east-1f	subnet-ef224fe0	172.31.48.0/20	

[Cancel](#) [Save](#)

Figure 37: AWS Aurora DB

Once done, click on Save button. This action will save the settings and will take you back to the Advance Configuration page (see [Figure 37](#)).

5.5.1.2.7 Step 7: Launch Application

These settings done so far are sufficient to create a High Availability Load Balanced application. Click on `Create App` button (see [Figure 37](#)).

This will initiate the process of application creation. Once the process completes, the application health will be displayed with a green check and the application URL (which points to the public load balancer) will be displayed in the top section (see [Figure 38](#)).

Recent Events

Time	Type	Details
2019-02-17 15:01:54 UTC-0500	INFO	Successfully launched environment: Mytestapplication-env
2019-02-17 15:01:54 UTC-0500	INFO	Application available at Mytestapplication-env.wtpep7ypvp.us-east-1.elasticbeanstalk.com.
2019-02-17 15:01:42 UTC-0500	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 17 seconds ago and took 3 minutes.
2019-02-17 15:01:08 UTC-0500	INFO	Created Load Balancer listener named: am.aws.elasticloadbalancing.us-east-1.375283443325.listener/app/wseb-AWSEB-EUFES12NCE2K/4b550aef178edcd9/a024664f81f72b14
2019-02-17 15:00:52 UTC-0500	INFO	Created load balancer named: am.aws.elasticloadbalancing.us-east-1.375283443325.loadbalancer/app/wseb-AWSEB-EUFES12NCE2K/4b550aef178edcd9

Figure 38: AWS Beanstalk

5.5.1.2.8 Step 8: Verify application URL

Click on the URL generated in the previous step. This will open the homepage of the sample application (see [Figure 39](#)).

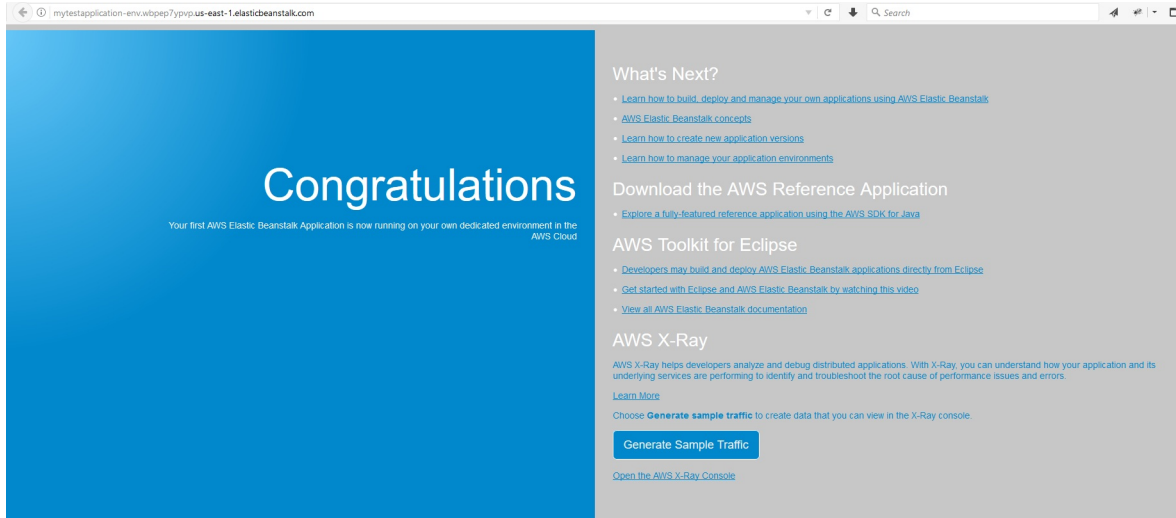


Figure 39: AWS Beanstalk

5.5.1.3 Explore Amazon Elastic Beanstalk Features

5.5.1.3.1 Beanstalk Deployment

Beanstalk deployment is quick and easy. Lets deploy a sample springboot war to this environment. Go to the application dashboard. Click on `Upload and Deploy` button. This will open a pop up that will allow to upload package and apply a version label. One can also select to deploy to all instances at once or batch wise. Click on deploy once ready (see [Figure 40](#)).

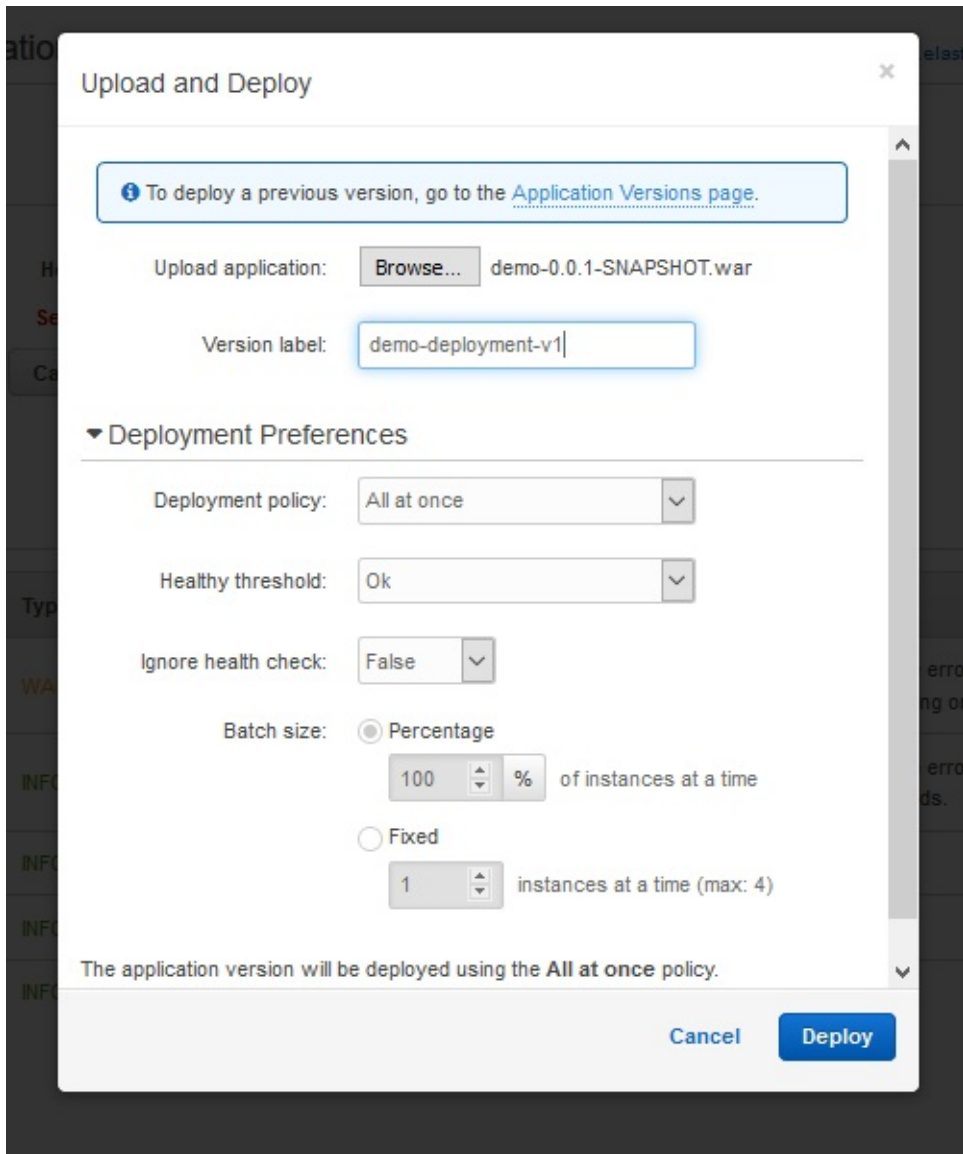


Figure 40: AWS Beanstalk

Upon completion of the deployment, beanstalk will display success message in the recent events sections (see [Figure 41](#)).

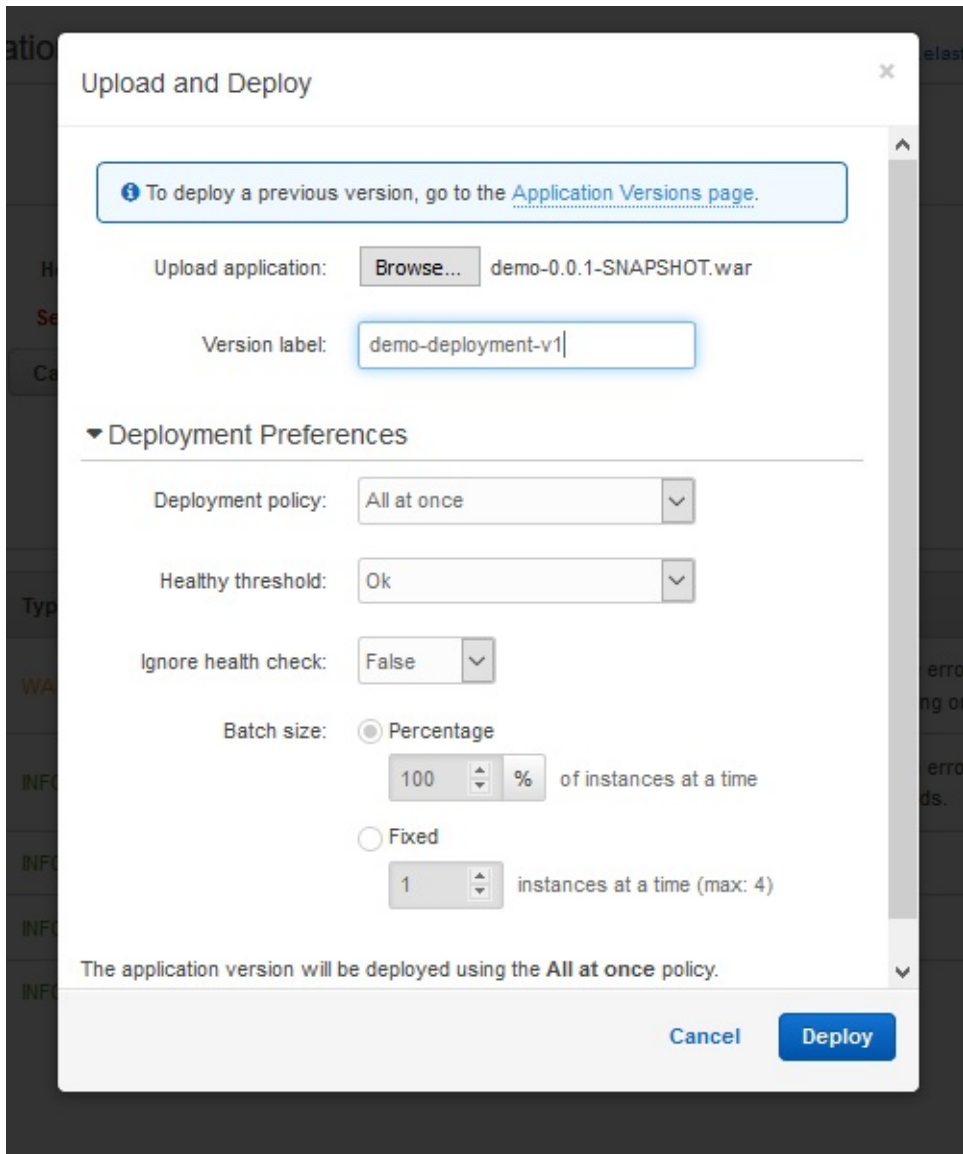


Figure 41: AWS Beanstalk

The updated application can be validated by clicking the beanstalk URL (see [Figure 42](#)).

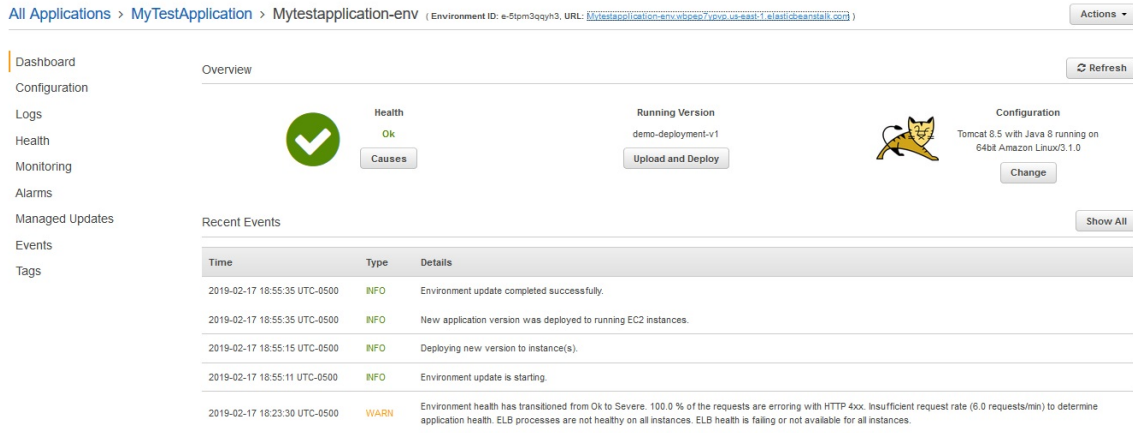


Figure 42: AWS Beanstalk

5.5.1.3.1.1 Tracking Deployments

All deployments can be tracked by clicking the [Application Versions](#) page on the upload pop up (see [Figure 40](#)).

Once you click the link, a new page is opened which allows to: * View and download all past packages deployed. * Deploy a previous package again * Manage the life cycle policy for backing up past deployments (see [Figure 43](#)).

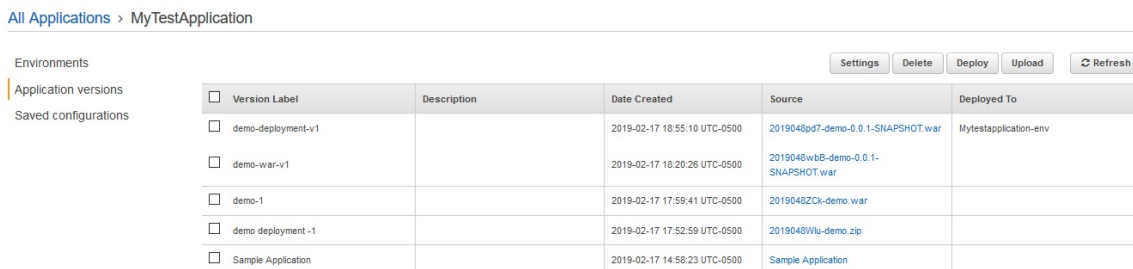


Figure 43: AWS Beanstalk

5.5.1.3.2 Beanstalk Env management

The Action button on the application dashboard allows to perform the next set of actions on the application environment:

- Restart App servers
- Save the configuration for re-use

- Load a new configuration
- Clone the environment
- Rebuild the environment
- Terminate the environment

(see [Figure 44](#))

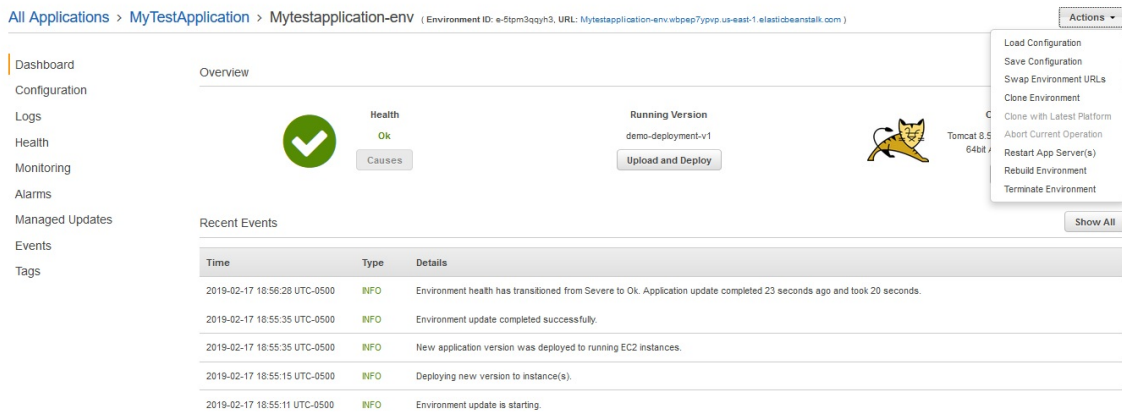


Figure 44: AWS Beanstalk

5.5.1.3.3 Beanstalk Logs

The application logs can be accessed from the Logs link in the application dashboard (see [Figure 45](#)).

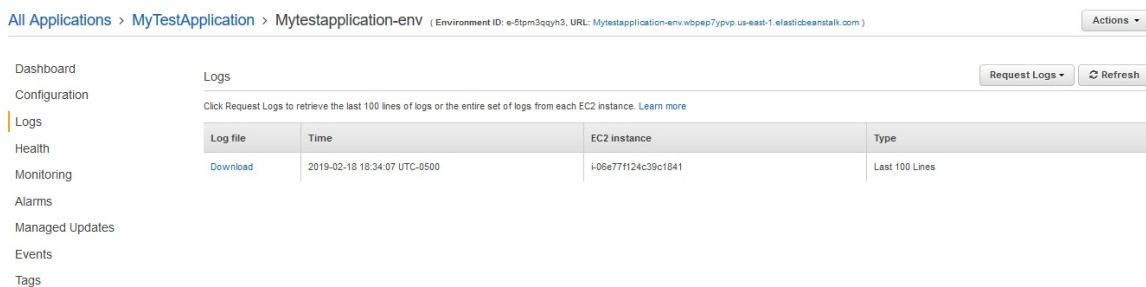


Figure 45: AWS Beanstalk

5.5.1.3.4 Beanstalk Instance Health

The Health link in the application dashboard takes to the health monitoring screen and allows to view the health of all the EC2 server instances associated with the beanstalk application.

Additionally, this screen gives option to reboot or terminate the selected EC2 instances (see [Figure 46](#)).

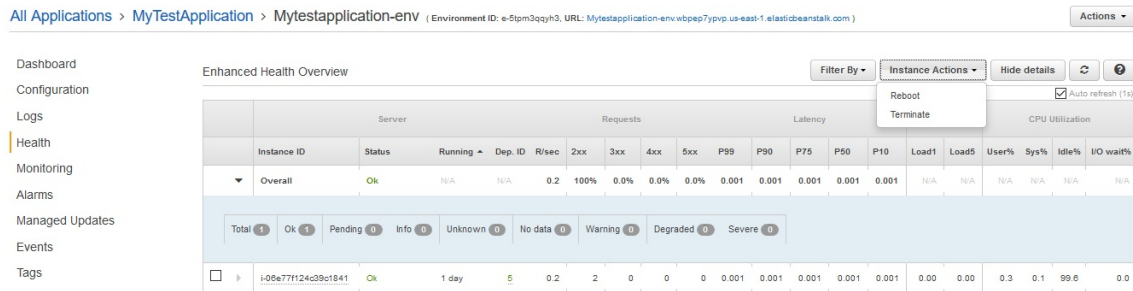


Figure 46: AWS Beanstalk

5.5.1.3.5 Beanstalk Environment Monitoring and Alarms

The Monitoring link in the application dashboard takes to the environment monitoring screen which displays different types of graphs / metrics under different sections. The duration and intervals for monitoring can be edited to adjust the granularity (see [Figure 47](#)).

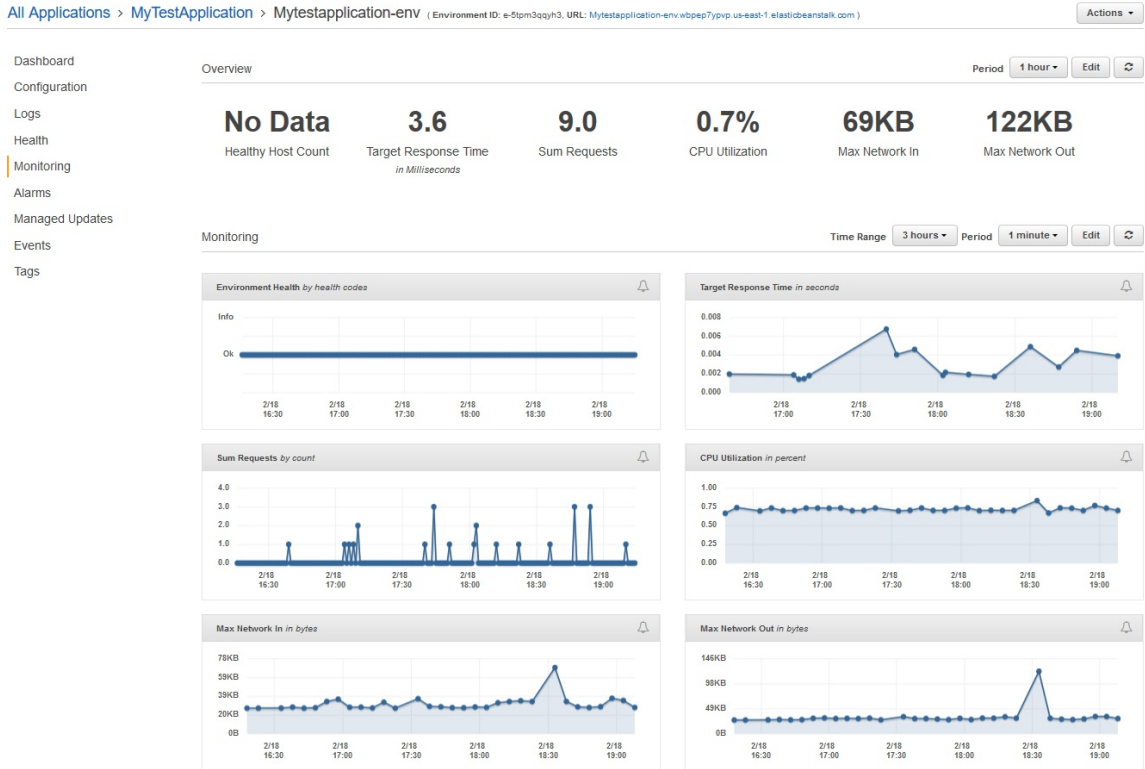


Figure 47: AWS Beanstalk

Each metric section also has a bell icon on the top right corner. Using this icon one can set alarms based on specific thresholds. Alarms can be integrated with SNS topics for notification or DynamoDB (see [Figure 48](#)).

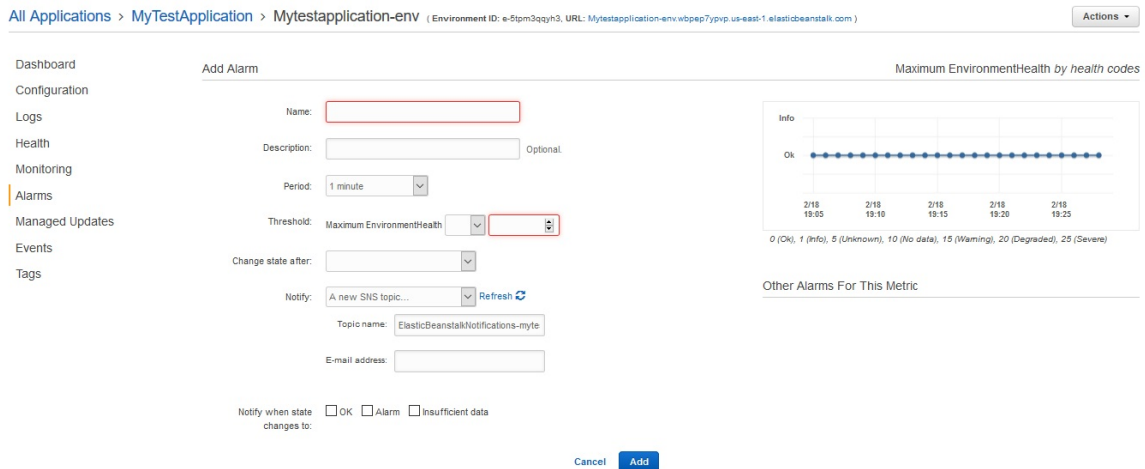


Figure 48: AWS Beanstalk

5.5.1.3.6 Beanstalk Patching Logs

The Managed Updates link in the application dashboard allows to view the history of the patching activities performed on the environment by AWS (see [Figure 49](#)).

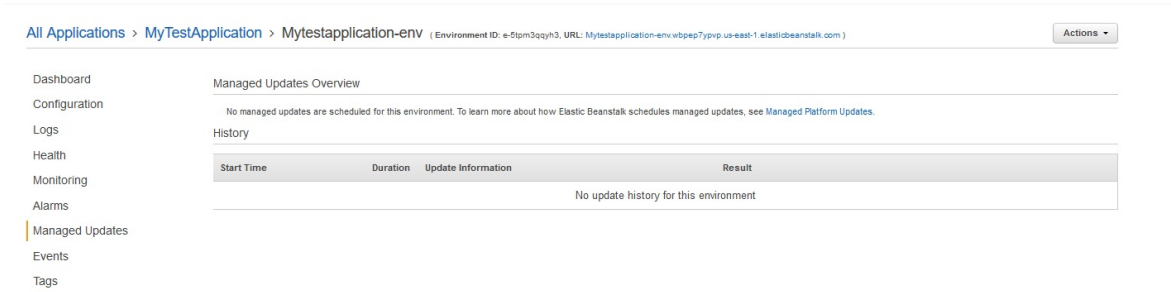


Figure 49: AWS Beanstalk

5.5.1.3.7 Beanstalk Event Logs

The Events link in the application dashboard allows to view the history of events occurred on the environment. These are typically from the configuring changes made, instances transitioning from one health status to another, or deployments made on the environment (see [Figure 50](#)).

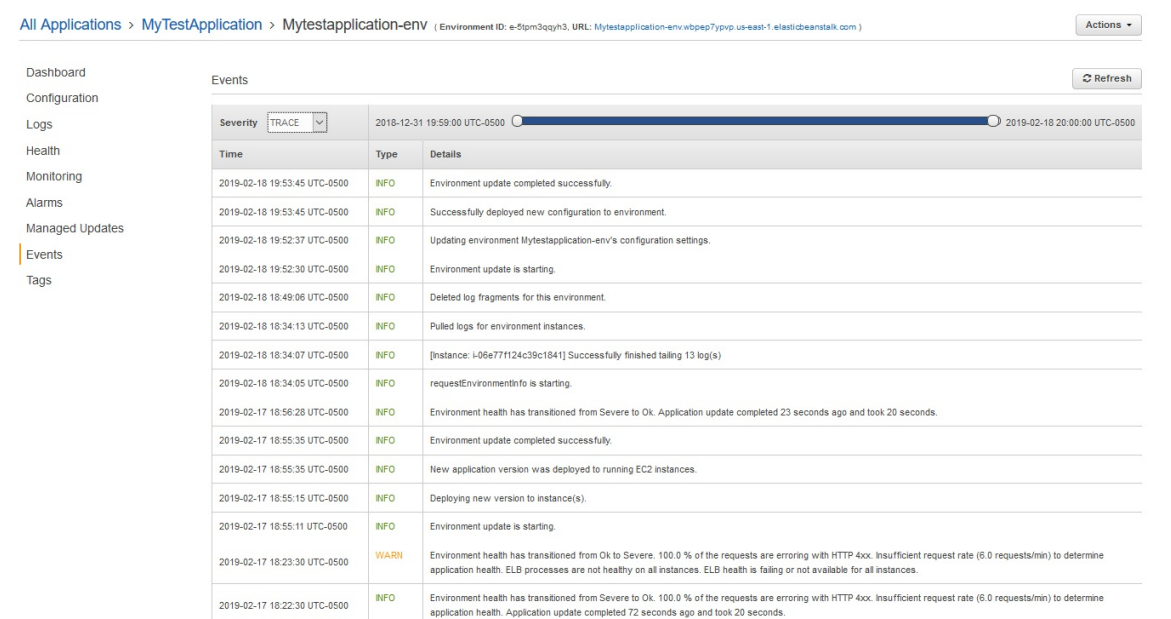


Figure 50: AWS Beanstalk

5.5.1.3.8 Beanstalk Tags

The Tags link in the application dashboard allows to view all the tags applied to

the environment. Tags are key value pairs that help to identify the beanstalk resources for management or pricing activities (see [Figure 51](#)).

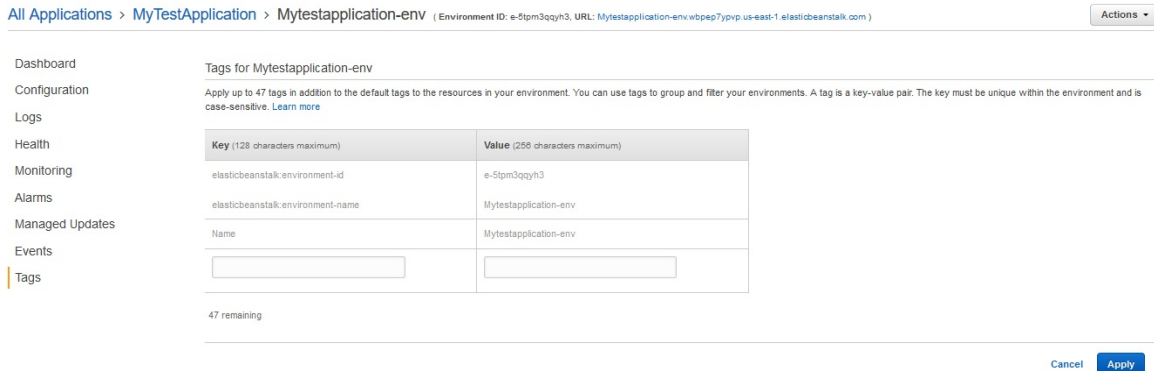


Figure 51: AWS Beanstalk

5.5.1.3.9 Beanstalk Configuration Management

The Configurations link in the application dashboard allows to modify various configurations for the beanstalk application. These are grouped into different sections which are explained under the following pages (see [Figure 52](#)).

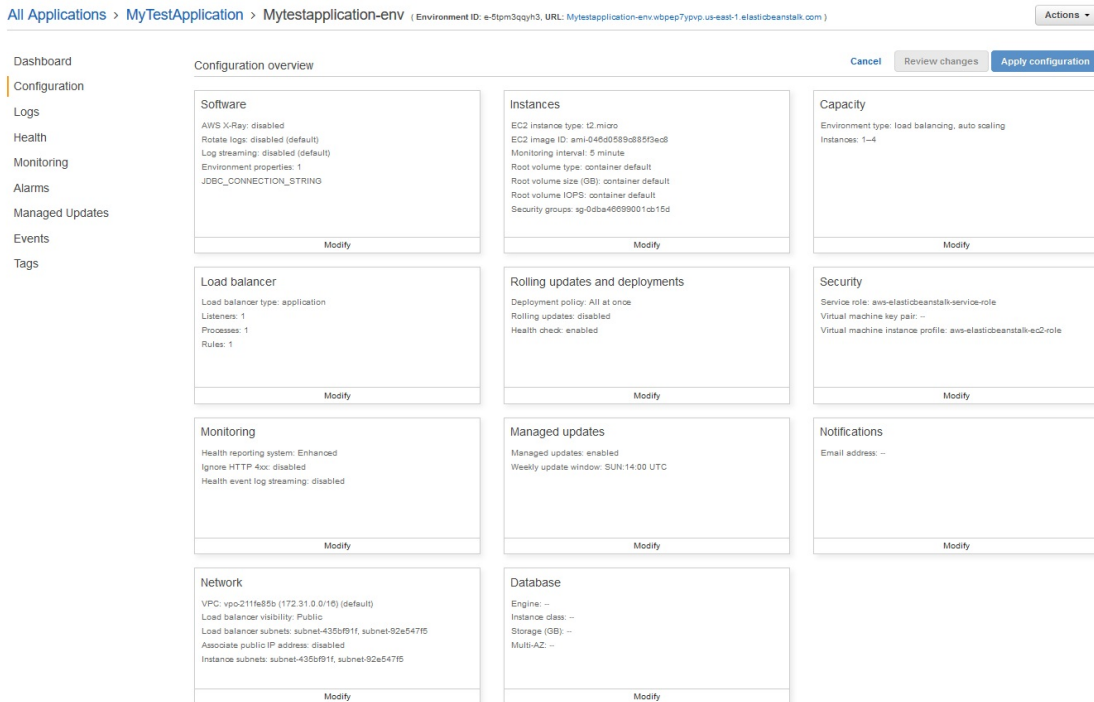


Figure 52: AWS Beanstalk

5.5.1.3.9.1 Software

5.5.1.3.9.1.1 Container Options

We can also configure the container settings to host the application. Below options are available for selection.

- Proxy Server - Apache, Nginx
- Gzip Compression
- Initial JVM Heap Size
- Max JVM Heap Size
- Max Perm Size

5.5.1.3.9.1.2 X Ray

AWS X Ray is a service that can be enabled to trace an application request end to end through to the underlying components. This is extremely useful in helping debug systems following distributed architecture. This service however is not free and will incur additional charges.

5.5.1.3.9.1.3 S3 Log Storage

We can also optionally rotate logs to Amazon S3. Again, S3 storage will incur additional charges.

5.5.1.3.9.1.4 Environment properties

The properties defined here are passed to the application as environment properties. These are especially useful in maintaining variables like the Database connection string which would vary between the development, staging and production environment.

Maintaining these properties here allows migration of same code package between environments without any code changes (see [Figure 53](#)).

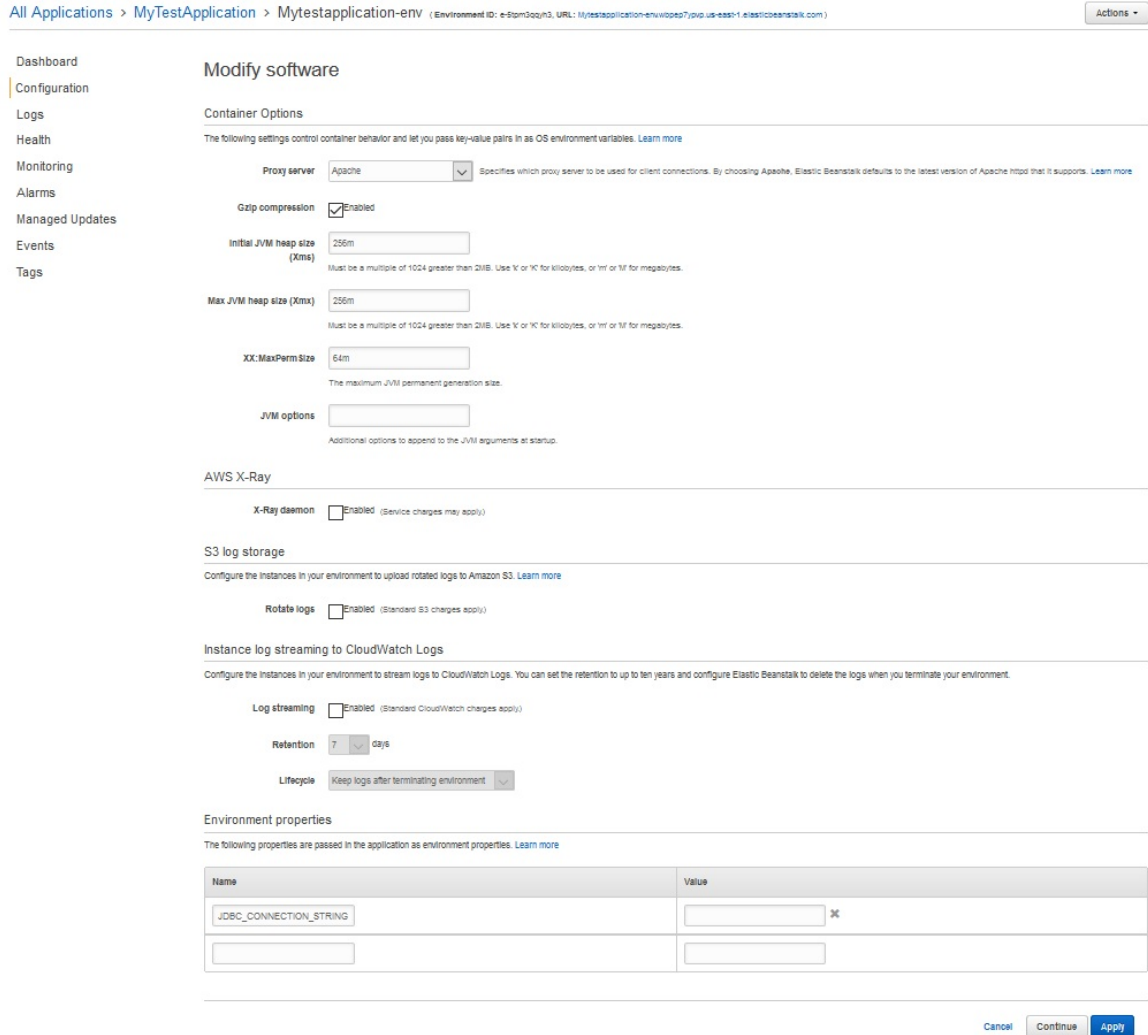


Figure 53: AWS Beanstalk

5.5.1.3.9.2 Instances

5.5.1.3.9.2.1 Instance Type

In case of fluctuating business needs, the EC2 server instance type used to host the application can also be changed. EC2 instance come in various categories (t2,t3, m4, m5, c1, c2 etc.) and size (micro, small, medium, large etc) and can be selected based on the application needs and usage pattern.

5.5.1.3.9.2.2 Amazon CloudWatch monitoring

The cloudwatch monitoring period can be selected between 1 and 5 mins. Smaller monitoring interval incur additional charges. The default monitoring

window is 5 mins.

5.5.1.3.9.2.3 Root volume (boot device)

Beanstalk also provides flexibility to choose the volume type (Magnetic, General Purpose SSD, Provisioned IOPS SSD), size and IOPS for the instances.

5.5.1.3.9.2.4 Security Groups

Security groups mapped to the EC2 instances allow to define rules restrict traffic ingress (see [Figure 54](#)).

Dashboard
Configuration
Logs
Health
Monitoring
Alarms
Managed Updates
Events
Tags

Modify instances

Instance type
Choose an instance type that best matches your workload requirement.

Instance type:

AMI ID:

Amazon CloudWatch monitoring
The time interval between when metrics are reported from the EC2 instances.

Monitoring interval:

Root volume (boot device)

Root volume type:

Size: GB
The number of gigabytes of the root volume attached to each instance.

IOPS: IOPS
Input/output operations per second for a provisioned IOPS (SSD) volume.

EC2 security groups

	Group name	Group ID	Name
<input type="checkbox"/>	awseb-e-5tpm3qzyh3-stack-AWSEBLoadBalancerSecurityGroup-1381KD46GGJQD	sg-07d60198a bd480ea1	Mytestapplication-env
<input checked="" type="checkbox"/>	awseb-e-5tpm3qzyh3-stack-AWSEBSecurityGroup-HOP64VYSNCC6	sg-0dba46699 001cb15d	Mytestapplication-env
<input type="checkbox"/>	default	sg-68a78c23	
<input type="checkbox"/>	launch-wizard-1	sg-0344a0de1 a7909ef3	EC2-Tarun
<input type="checkbox"/>	rds-launch-wizard	sg-0317860e6 a60a1f0d	

Figure 54: AWS Beanstalk

5.5.1.3.9.3 Capacity

5.5.1.3.9.3.1 Auto Scaling Group

Auto Scaling Groups allow to define an auto scaling policy where in the beanstalk environment would toggle between min and max number of instances defined based on scaling triggers like

- Network Out
- CPU Utilization
- Network In
- Disk IO
- Latency
- Healthy Host Count
- Unhealthy Host Count
- Response Time

Alternatively, auto scaling can be timed if we know the time period of peak load (see [Figure 55](#)).

All Applications > MyTestApplication > Mytestapplication-env (Environment ID: s-9pm3ppn2, UK-1: Mytestapplication-env-rtb2k7jpp-ua-est-1-elasticbeanstalk.com) Actions

Dashboard
 Configuration
 Logs
 Health
 Monitoring
 Alarms
 Managed Updates
 Events
 Tags

Modify capacity

Auto Scaling Group

Configure the compute capacity of your environment and Auto Scaling settings to optimize the number of instances used.

Environment type: Load balanced

Instances: Min 1, Max 4

Availability Zones: Any
Number of Availability Zones (AZs) to use.

Placement: us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, us-east-1f
Specify Availability Zones (AZs) to use.

Scaling cooldown: 360 seconds

Scaling triggers

Metric: NetworkOut
Change the metric that is monitored to determine if the environment's capacity is too low or too high.

Statistic: Average
Choose how the metric is aggregated.

Unit: Bytes

Period: 5 Min
The period between metric evaluations.

Breach duration: 5 Min
The amount of time a metric can exceed a threshold before triggering a scaling operation.

Upper threshold: 6000000 Bytes

Scale up Increment: 1 EC2 Instances

Lower threshold: 2000000 Bytes

Scale down Increment: -1 EC2 Instances

Time-based Scaling

Use the following settings to control time-based scaling actions. [Learn more](#)

Current status: 1 Instance(s) in service, Min: 1, Max: 4

Time zone: UTC Local Actions Add scheduled action

<input type="checkbox"/>	Name	Min	Max	Desired	Next occurrence (UTC)
No scheduled actions					

Cancel Continue Apply

Figure 55: AWS Beanstalk

5.5.1.3.9.4 Load Balancer

Next we will discuss the options to modify the load balance configurations:

- Add remove http / https listeners
- Serve http / https services on one or more ports
- Health check paths

Opening application on https will require you to upload signed certificates (see [Figure 56](#)).

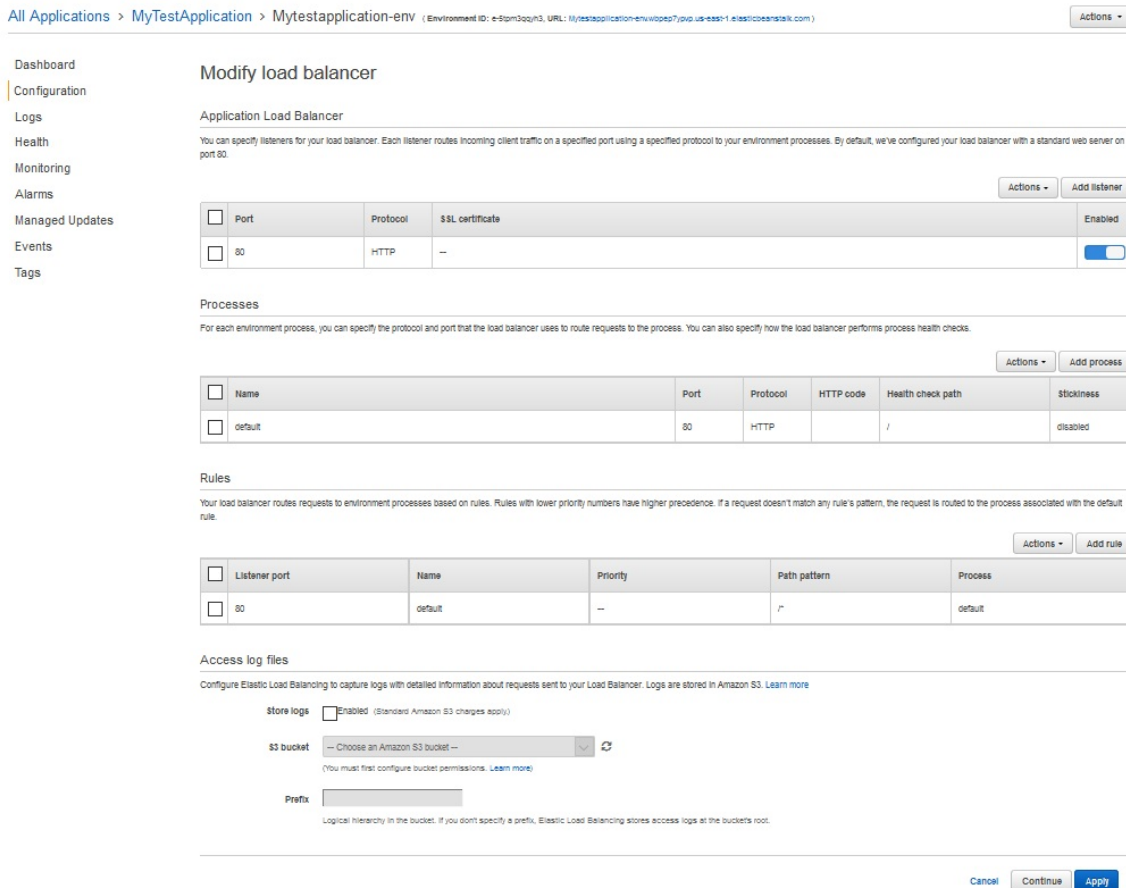


Figure 56: AWS Beanstalk

5.5.1.3.9.5 Rolling updates and deployments

5.5.1.3.9.5.1 Application Deployments

Beanstalk allows to specify if the deployments will be done on all instances at once or one by one. This helps ensuring at least one application node is up while the other is being deployed with the new code.

5.5.1.3.9.5.2 Configuration Updates

Similar to application deployments, we can also specify if the configuration changes on VMs will be done on all instances at once or one by one. (see [Figure 57](#)).

All Applications > MyTestApplication > Mytestapplication-env (Environment ID: e-0pm3qqn3, URL: http://mytestapplication-env.us-east-1.elasticbeanstalk.com) Actions -

Dashboard
Configuration
Logs
Health
Monitoring
Alarms
Managed Updates
Events
Tags

Modify rolling updates and deployments

Application deployments

Choose how AWS Elastic Beanstalk propagates source code changes and software configuration updates. [Learn more](#)

Deployment policy: All at once

Batch size: Percentage (100 % of the fleet at a time) Fixed (1 Instances at a time)

Configuration updates

Changes to virtual machine settings and VPC configuration trigger rolling updates to replace the instances in your environment without downtime. [Learn more](#)

Rolling update type: Disabled

Batch size: 1 (The maximum number of instances to replace in each phase of the update.)

Minimum capacity: 1 (The minimum number of instances to keep in service at all times.)

Pause time: 0 Hour, 5 Minutes, 30 Seconds (Pause the update for up to an hour between each batch.)

Deployment preferences

Customize health check requirements and deployment timeouts.

Ignore health check: Don't fail deployments due to health check failures.

Healthy threshold: Ok (Lower the threshold for an instance in a batch to pass health checks during an update or deployment.)

Command timeout: 600 (Change the amount of time in seconds that AWS Elastic Beanstalk allows an instance to complete deployment commands.)

Cancel Continue Apply

Figure 57: AWS Beanstalk

5.5.1.3.9.6 Security

The security section has options to map the following list of attributes:

- Service Role - AWS role that beanstalk will assume. This role specifies the permissions for beanstalk to provision resources like EC2.
- Instance Role - AWS role that the EC2 instances will assume. This role specifies the permissions to EC2 instances to access other AWS services.

- EC2 key pair - This selected key pair can be used to SSH into the EC2 instances if needed (see [Figure 58](#)).

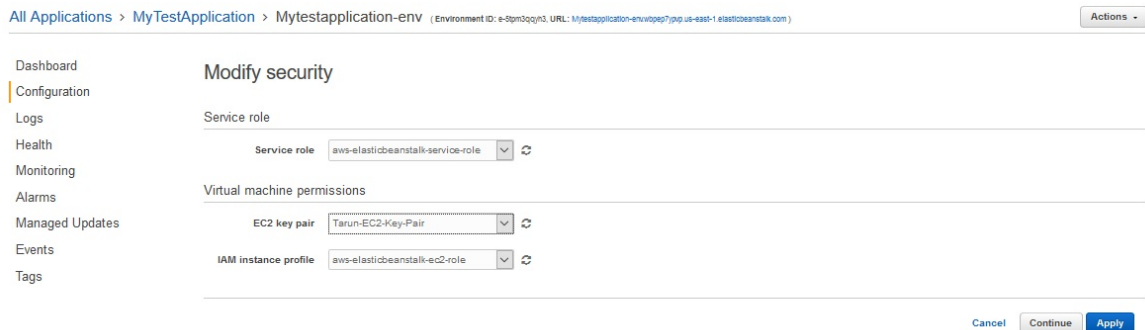


Figure 58: AWS Beanstalk

5.5.1.3.9.7 Monitoring

Next we will discuss the monitoring options available in AWS Beanstalk.

5.5.1.3.9.7.1 Health Reporting

User can select between basic and enhanced monitoring options. While enhanced monitoring incurs additional charges, it allows to select custom metrics to be reported to Cloud Watch.

5.5.1.3.9.7.2 Health monitoring rule customization

We can choose to ignore http 4XX errors for monitoring health checks on need basis. This is helpful in cases where the application by design returns 4XX error codes based on certain conditions(see [Figure 59](#)).

All Applications > MyTestApplication > Mytestapplication-env (Environment ID: e-5pm3oqj3, URL: Mytestapplication-envops?pg.us-east-1.elasticbeanstalk.com) Actions

Dashboard
Configuration
 Logs
 Health
 Monitoring
 Alarms
 Managed Updates
 Events
 Tags

Modify monitoring

Health reporting

Enhanced health reporting provides free real-time application and operating system monitoring of the instances and other resources in your environment. The `EnvironmentHealth` custom metric is provided free with enhanced health reporting. Additional charges apply for each custom metric. For more information, see [Amazon CloudWatch Pricing](#).

System Enhanced
 Basic

CloudWatch Custom Metrics	Instance	Environment
	ApplicationLatencyP10	ApplicationLatencyP10
	ApplicationLatencyP50	ApplicationLatencyP50
	ApplicationLatencyP75	ApplicationLatencyP75
	ApplicationLatencyP85	ApplicationLatencyP85
	ApplicationLatencyP90	ApplicationLatencyP90
	ApplicationLatencyP95	ApplicationLatencyP95
	ApplicationLatencyP99	ApplicationLatencyP99
	ApplicationLatencyP99.9	ApplicationLatencyP99.9
	ApplicationRequests2xx	ApplicationRequests2xx
	ApplicationRequests3xx	ApplicationRequests3xx
	ApplicationRequests4xx	ApplicationRequests4xx
	ApplicationRequests5xx	ApplicationRequests5xx
	ApplicationRequestsTotal	ApplicationRequestsTotal
	CPUIdle	InstancesDegraded
	CPUWait	InstancesInfo
	CPUUtil	InstancesNoData
	CPUNice	InstancesOk
	CPUSoftirq	InstancesPending
	CPUSystem	InstancesSevere
	CPUUser	InstancesUnknown
	InstanceHealth	InstancesWarning
	LoadAverage1min	
	LoadAverage5min	
	RootFilesystemUtil	

Hold the Ctrl/Command key while clicking to select multiple metrics.

Health monitoring rule customization

Configure the HTTP application status codes included in determining your environment's health. [Learn more](#)

Ignore HTTP 4xx Enabled

Health event streaming to CloudWatch Logs

Configure Elastic Beanstalk to stream environment health events to CloudWatch Logs. You can set the retention up to a maximum of ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.

Log streaming Enabled (Standard CloudWatch charges apply)

Retention days

Lifecycle

Cancel Continue Apply

ic-pact-1
 Figure 59: AWS Beanstalk

5.5.1.3.9.8 Managed Updates

We can also optionally configure the automatic updates to the platform based on specific time period / slot for updates (see [Figure 60](#)).

All Applications > MyTestApplication > Mytestapplication-env (Environment ID: e-5pm3oqj3, URL: Mytestapplication-envops?pg.us-east-1.elasticbeanstalk.com) Actions

Dashboard
Configuration
 Logs
 Health
 Monitoring
 Alarms
 Managed Updates
 Events
 Tags

Modify managed updates

Managed platform updates

Enable managed platform updates to apply platform updates automatically during a weekly maintenance window that you choose. Your application stays available during the update process.

Managed updates Enabled

Weekly update window at : UTC
Any available managed updates will run between Sunday 9:00 AM and Sunday 11:00 AM (-0500 GMT).

Update level

Instance replacement If enabled, an Instance replacement will be scheduled if no other updates are available.

Cancel Continue Apply

Figure 60: AWS Beanstalk

5.5.1.3.9.9 Notifications

Using this section, we can specify email address to receive notifications about important events (see [Figure 61](#)).

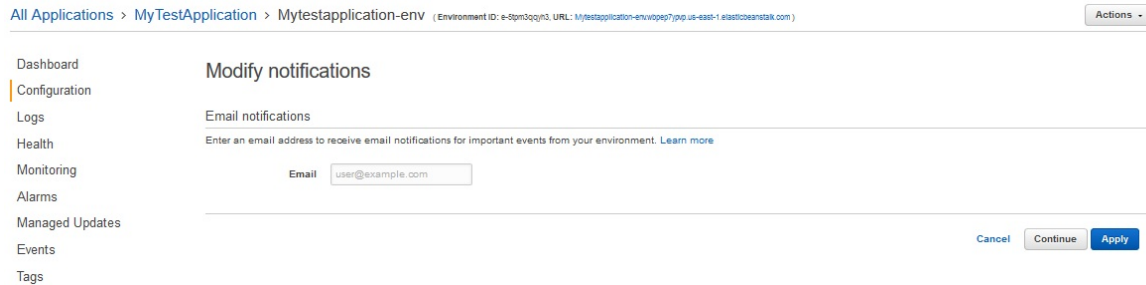


Figure 61: AWS Beanstalk

5.5.1.3.9.10 Network

We also have an option to update the subnets associated with the load balancer and EC2 instances at any time.

Note : VPC on the application can be mapped only at the time of creation of the beanstalk application and cannot be modified later (see [Figure 62](#)).

All Applications > MyTestApplication > Mytestapplication-env (Environment ID: e-5pm13oq/n3, URL: Mytestapplication-env/w8pe7jv0.us-east-1.elasticbeanstalk.com) Actions ▾

Dashboard

Configuration

Logs

Health

Monitoring

Alarms

Managed Updates

Events

Tags

Modify network

Virtual private cloud (VPC)

VPC vpc-21f8559 (172.31.0.0/16)

Load balancer settings

Assign your load balancer to a subnet in each Availability Zone (AZ) in which your application runs. For a publically accessible application, set Visibility to Public and choose public subnets.

Visibility ▾

Make your load balancer Internal if your application serves requests only from connected VPCs. Public load balancers serve requests from the Internet.

Load balancer subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-4359f91f	172.31.32.0/20	
<input checked="" type="checkbox"/>	us-east-1b	subnet-92e547f5	172.31.0.0/20	
<input type="checkbox"/>	us-east-1c	subnet-6e03af40	172.31.80.0/20	
<input type="checkbox"/>	us-east-1d	subnet-0f507145	172.31.16.0/20	
<input type="checkbox"/>	us-east-1e	subnet-3e690000	172.31.64.0/20	
<input type="checkbox"/>	us-east-1f	subnet-ef224f60	172.31.48.0/20	

Instance settings

Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances.

Public IP address Assign a public IP address to the Amazon EC2 instances in your environment.

Instance subnets

	Availability Zone	Subnet	CIDR	Name
<input checked="" type="checkbox"/>	us-east-1a	subnet-4359f91f	172.31.32.0/20	
<input checked="" type="checkbox"/>	us-east-1b	subnet-92e547f5	172.31.0.0/20	
<input type="checkbox"/>	us-east-1c	subnet-6e03af40	172.31.80.0/20	
<input type="checkbox"/>	us-east-1d	subnet-0f507145	172.31.16.0/20	
<input type="checkbox"/>	us-east-1e	subnet-3e690000	172.31.64.0/20	
<input type="checkbox"/>	us-east-1f	subnet-ef224f60	172.31.48.0/20	

Figure 62: AWS Beanstalk

5.5.1.3.9.11 Database

If required, we can also create and manage a RDS database as part of the beanstalk configuration (see [Figure 63](#)).

All Applications > MyTestApplication > Mytestapplication-env (Environment ID: e-5ipm3qyq3, URL: Mytestapplication-envw0pep7jvvp.us-east-1.elasticbeanstalk.com) Actions

Dashboard

Configuration

Logs

Health

Monitoring

Alarms

Managed Updates

Events

Tags

Modify database

Add an Amazon RDS SQL database to your environment for development and testing. AWS Elastic Beanstalk provides connection information to your instances by setting environment properties for the database hostname, username, password, table name, and port. When you add a database to your environment, its lifecycle is tied to your environments. For production environments, you can configure your instances to connect to a database. [Learn more](#)

Restore a snapshot

Restore an existing snapshot in your account, or create a new database.

Snapshot ↕

Database settings

Choose an engine and instance type for your environment's database.

Engine

Engine version

Instance class

Storage GB

Choose a number between 5 GB and 1024 GB.

Username

Password

Retention

When you terminate your environment, your database instance is also terminated. Choose Create snapshot to save a snapshot of the database prior to termination. Snapshots incur standard storage charges.

Availability

Cancel Continue Apply

Figure 63: AWS Beanstalk

5.5.1.4 Access Beanstalk using python

The beanstalk application and all of its associated resources can also be accessed and managed programatically using boto package in python.

Here is the link that describes the boto methods available for beanstalk.

- <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/service/>

5.5.1.5 Exercises

ElasticBeanstalk.1:

Follow steps defined in the sections above to set up a beanstalk application in Python using default sample application provided by AWS. Use the generated beanstalk URL to verify that the application is accessible.

ElasticBeanstalk.2:

Once you are able to access the beanstalk application using the generated URL, deploy your custom application and validate if the environment reflects the changes as per your new deployment package.

ElasticBeanstalk.3:

Click on `configuration` link on left hand menu in the beanstalk console. Next click on the `capacity` link and set up an Auto Scaling policy to keep the application running on a minimum of 2 servers at all times. Save the new configuration and allow beanstalk to install the changes to the environment. Verify that the application is now running on 2 servers. This can be verified by clicking on the `Health` link.

ElasticBeanstalk.4:

Now select 1 of the 2 servers and terminate it from the `Instance Actions` button on the top right on `Health Page`. Wait for some time to confirm if beanstalk automatically adds a new server instance to match the minimum threshold set up in exercise 3.

ElasticBeanstalk.5:

Click on `configuration` link and open the `Load Balancer` section. Add new rules to make the application listen on http / https on ports other than the default 80 and 443 ports. Validate that the application URLs work on the new ports once the changes are applied.

5.5.1.6 References

- <https://aws.amazon.com/elasticbeanstalk/> [23]

5.5.2 Compliance and Grid Computing



5.5.3 Visualization

0

All tools must be free.

5.5.3.1 Chart Types

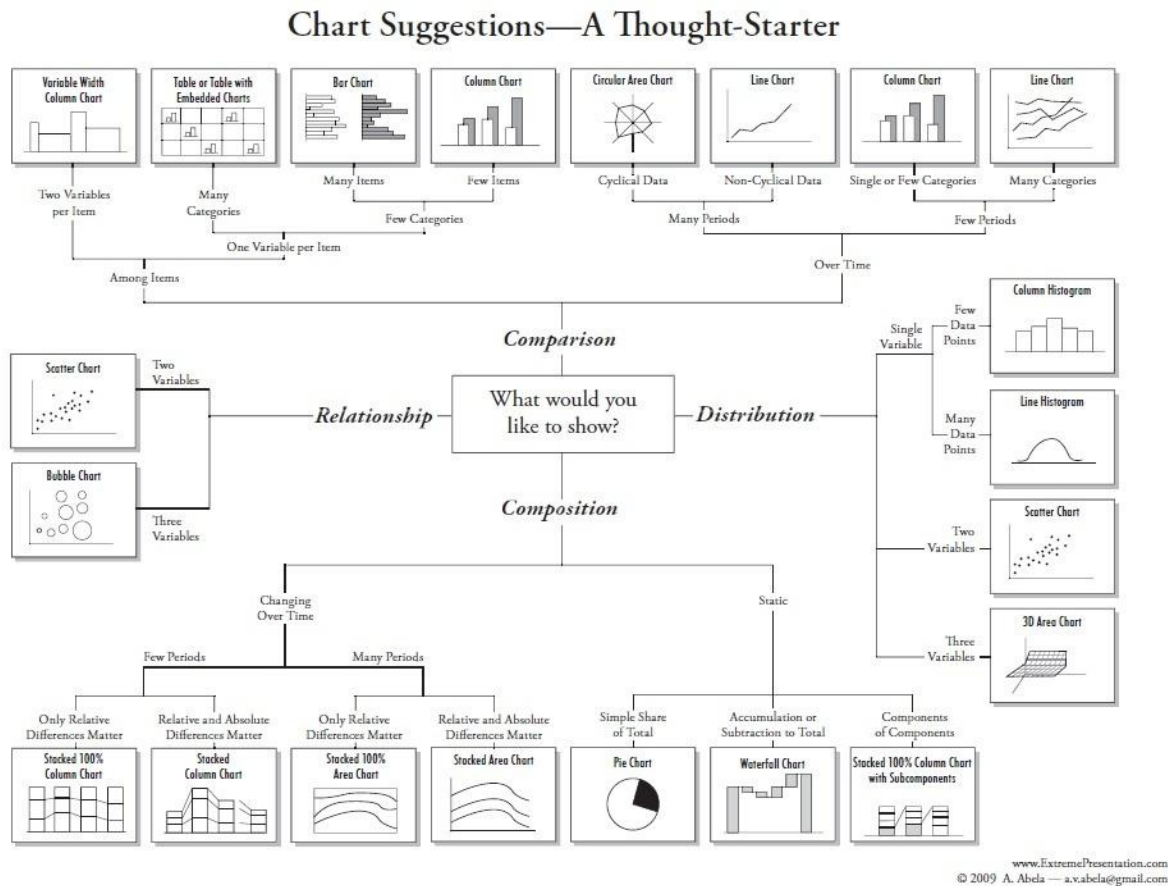


Figure 64: Visualisation

- <https://extremepresentation.typepad.com/files/choosing-a-good-chart-09.pdf>

5.5.3.2 D3 Gallery

- <https://github.com/d3/d3/wiki/Gallery>
- <https://c3js.org/examples.html>
- <http://nvd3.org/examples/index.html>

5.5.3.3 Matplot Gallery

TBD

5.5.3.4 Bokeh Gallery

- <https://bokeh.pydata.org/en/latest/docs/gallery.html>

5.5.3.5 R Gallery

- <https://www.r-graph-gallery.com/>

5.5.3.6 Gnuplot

- <http://www.gnuplot.info/screenshots/index.html#demos>

5.5.3.7 React

- <https://uber.github.io/react-vis/examples/showcases/plots>
- <https://formidable.com/open-source/victory/gallery/>

5.5.3.8 Tableau

Tableau is not reproducible and creates very poor print quality images. YOU can use it to explore data, but must use another tool to have reproducible images.

5.5.4 Distributed Message Queues

5.5.4.1 AMPQ

- https://en.wikipedia.org/wiki/Advanced_Message_Queueing_Protocol

Slides:

- <http://www.rabbitmq.com/resources/f2f-overview12.ppt>

Python:

Celery Distributed Task Queues

- <http://www.celeryproject.org/>

Message Broker:

RabbitMQ

Tutorial:

- <http://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.html>

Comparision

- <http://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.html>
- <http://queues.io/>
- <http://bravenewgeek.com/dissecting-message-queues/>

Kafka:


NATS?

5.5.4.2 screenshot rename automator 

- <https://blog.prototypr.io/organize-your-screenshots-like-a-boss-6be10e20d6a1>

6 ISSUES

6.1 GITHUB ISSUES

 *The issues are automatically created from Github Issues. Please change them directly in github.*

Do not modify the table. or this file

To file new issues, please go to:

- <https://github.com/cloudmesh-community/book/issues>

Additionally, we use the following notation to coordinate sections and chapters that are open, conducted by students. This is indicated by a prefix to the issue summary.

Prefix	Description
Internal:	Assigned to a staff member and done internally by them
Open:	A task is open and can be executed also by students.
Assigned:	A task is assigned and is currently been executed by the assignee
Done:	The task is done and needs review.

Assignees can change the summary to done. We will experiment with possibly additional labels to communicate the state via labels. Please stay tuned.

6.1.1 Internal Issues

.

N	#	Title	Assignee	Labels
38	266	Internal: Github video use	tbalson	internal
47	254	Fingerprint matching	pulasthi	internal section
60	239	Internal: New trends	None	assigned internal
65	227	internal: Virtual machine .md file is missing	None	bug internal
69	180	E534 Cloud Section lecture videos merged into E-books	None	assigned internal
70	173	internal: docker: biber installation	None	internal
74	144	internal: bigdata missing improvements	laszewsk	internal
76	126	bibtex: 523	None	internal
77	77	add mapreduce to syllabus	laszewsk	assigned internal
78	76	openapi demo	laszewsk	assigned internal

6.1.2 Open Sections

N	#	Title	Assignee	Labels
10	334	MiniProject: Windows 10 compatibility	None	MiniProject chapter help wanted open section
11	333	MiniProject: conda package for cloudmesh	himanshu3jul	MiniProject chapter help wanted

				open section
15	320	internal: improve python code in AI	tbalson	e222 open section
16	319	OpenApi Tools	pulasthi	assigned open section
23	305	Nauta	None	open section
24	302	MongoDB	pulasthi	chapter open section
28	298	Pi: docker and grafana and influxdb	None	open section
40	261	Low Priority: Visual Studio for Cloud Computing (PyCharm is better)	None	e222 open section
43	258	Open: Apache Flink	None	chapter open section
48	253	Ubuntu on an USB stick for Windows 10	None	open section
49	252	Packer	None	open section
51	250	Fn Project	None	open section
52	249	IronFunctions	None	open section
53	248	Fission	None	open section
54	247	Riff	None	open section
56	245	Hadoop virtual cluster installation using cloudmesh	None	open section

58	243	Artificial intelligence service with REST	tbalson	assigned open section
64	232	Section: Port forwarding	ameet20	open section
66	223	Open: Section CircleCi	None	open section

6.1.3 Open Chapters

N	#	Title	Assignee	Labels
9	335	MiniProject: uwsgi	pulasthi	MiniProject chapter open
10	334	MiniProject: Windows 10 compatibility	None	MiniProject chapter help wanted open section
11	333	MiniProject: conda package for cloudmesh	himanshu3jul	MiniProject chapter help wanted open section
19	312	jupyterhub on kubernetes	None	chapter open
24	302	MongoDB	pulasthi	chapter open section
41	260	Open: Cloud foundry	None	chapter

				open
43	258	Open: Apache Flink	None	chapter open section

6.1.4 Assigned Sections

N	#	Title	Assignee	Labels
7	347	Section: AWS AuroraDB	trawat87	assigned section
8	337	MiniProject: Mongo install on Windows 10	himanshu3jul	MiniProject assigned section
14	327	Section: ISO 27001 Security Standards For Data Centers	robludwig	assigned section
16	319	OpenApi Tools	pulasthi	assigned open section
18	317	Pi: DIMM	None	assigned section
20	311	Avro	pulasthi	assigned section
21	310	HPC in the cloud	colliner	assigned section
25	301	Twister2	pulasthi	TA assigned section
31	292	Section: Infrastructure as Code - Terraform	joshish-iu	assigned section
32	286	Section: Data Services: AWS	joshish-iu	TA assigned

RedShift				section
34	275	Assigned: Amazon SageMaker	tvangalapat	TA assigned section
37	268	Chapter: Google Drive Python interface	g14uok	assigned chapter section
39	262	Open: Windows subsystems for Linux	None	assigned section
50	251	OpenLambda	garbeandy	assigned section
55	246	OpenFaaS	mallik3006	assigned section
57	244	Google IaaS Cloud Services	kelifine	assigned section
58	243	Artificial intelligence service with REST	tbalson	assigned open section
59	242	Scikit-Learn	tbalson	assigned section
63	233	Section: AWS DocumentDB	trawat87	assigned section

6.1.5 Assigned Chapters

N	#	Title	Assignee	Labels
6	353	Chapter : AWS Beanstalk	trawat87	assigned chapter
33	282	Assigned: Chapter on Apache	anthonyduer	TA assigned

Spark				chapter
35	273	Assigned: Databricks	himanshu3jul	TA assigned chapter
36	272	Chapter: Puppet	bfeng	TA assigned chapter
37	268	Chapter: Google Drive Python interface	g14uok	assigned chapter section
42	259	Chapter: Box	kelifine	assigned chapter
44	257	Apache Avro (Python)	pulasthi	assigned chapter
45	256	Assigned: Chapter & Sections: Scala for cloud computing	hrbahramian	assigned chapter
46	255	Julia	keithhickman08	TA assigned chapter
61	236	Open: EMR section(s)	bfeng	assigned chapter

6.1.6 Open Projects

N	#	Title	Assignee	Labels
62	234	Project: Virtual Cloud Directory	None	open project

6.1.7 Assigned Projects

.

N	\#	Title	Assignee	Labels
---	----	-------	----------	--------

.

7 WEEKLY

7.1 WEEKLY ACTIVITIES

In case you like to take the class with weekly activities you can look at the Syllabus table at the sections that are released for a particular date. The date means the activity is released on that date and you have time to conduct the activity.

7.1.1 Week 1: Activities Aug 26 - 30

In the first week we focus on the following activities:

1. Get familiar with the class and identify if you prefer to take the class in free form or in a more guided fashion. Contact us if you like to take it in more guided fashion via a private post to instructors. you will be able to use these weekly activity announcements to proceed. This weekly progress will be updated every Friday.
2. Understand that we have only 3 assignments. One of which is a significant project.
3. Download the Lecture notes ePub and install an ePub reader to read the lecture notes. Although we have a PDF version, this version is pretty large and will only be updated once a week. See in piazza the resource section to locate them
4. Learn piazza and do your bio post. Start a README.yml file (look at the assignment in piazza). Create a notebook.md file and keep it up to date on weekly basis.
5. Make sure to configure a computer on which you can do python 3.7.4. If you have not yet installed python we recommend you use pyenv. See our notes in the handbook about that. Pyenv allows you to install multiple python versions. Answer the question why anaconda is not a good version for python in the cloud. Explain why so many other courses recommend

you to install and use anaconda. Discuss this on piazza if you like. If you do not understand this or question this attend the online hour. If you have issues with this talk to the TA's.

6. Make sure you have a git client installed from which you can interact with git. Initially you can just use a web browser Find a spelling or grammar error in the lecture notes and correct it via the github.com Web browser while conducting a pull request.
7. Make sure to fill out the survey so we can create a github.com repository for your project
8. Start reviewing python. Especially do some language features as discussed in the notes and write a python module using setup.py and requirements.txt. Do a simple print hello world stat you install with `pip install .` . What is the difference to `pip install -e .` . How can you leverage this.

Remark: In this class everyone is allowed to help everyone. Each student will have if they do not participate in a group have a different assignment so cheating can be avoided. However if we detect that a student is not doing their work and it delegates all their work to other students this is considered cheating and an F will be assigned. Please use git commits. It is not sufficient if just one student of a group commits the entire project in case of group work.

7.1.2 Week 2: Aug 30 - Sep 6

7.1.2.1 Development machine

If you have not yet set up a computer with python 3.7.4 on it please do so. Remember you can use virtual machines and use virtualbox so you do not interfere with your base system or use a USB stick to boot into ubuntu. If you do not have a system work with the TAs to identify a solution that works for you

7.1.2.2 Data Center

Please look at the data center section (see @#sec:data-center) and read it. There are plenty of opportunities to contribute. Announce in piazza if you work on a

section so we minimize duplication of effort. Data Center

7.1.2.2.1 Datacenter Table

Find concrete evidence for data centers from industry and complete the table with the many question marks as much as possible. As the information may change over time, please include a year for which the data is valid, we may need to add a year column.

7.1.2.2.2 Data Center sections

As we got already some positive feedback about this section, I added some additional opportunities for more sections that can even evolve to a chapter if you like to focus on this.

7.1.2.2.3 Datacenter Exercises

Some of you asked what Exercises they can do for this week to do some work beyond reading the section. Please do not plagiarize. or just quote entire sections to avoid plagiarism. Work in a group to discuss and come up with high quality content.

The assignments are listed in Section **¿sec:exercises-energy?**

- E.Datacenter.2, A form link will be posted.
- E.Datacenter.3, A Form link will be posted.
- E.Datacenter.4

Please also pick from either assignments one. As this is a group assignment, please improve with the group.

- E.Datacenter.5
- E.Datacenter.6

7.1.2.3 Remarks: Sections

Although it is sufficient to just read the chapter we provide, its fun to do some google searches including just to look at images ... If you see something you

think we should add, propose a new section if you like. Please remember that you will need to do some sections that will be graded. We recommend that you contribute at least 5 sections. This is equivalent to one section every three weeks which is actually not much. Typically you will spend the first week researching and writing a draft. The second week experimentation or creating an example if applicable and the third week you will engage with other students and the TAs on reviews and improvements if needed.

7.1.2.4 Python till rest of the semester

Python 3.7.4: Set up a computer on which you can execute python 3.7.4. Some did not complete that task yet (see Section [¿sec:python-install?](#), Section [¿sec:python-intro?](#), Section [¿sec:python-language?](#)).

Review Python: start reviewing python. See our handbook. Some of the chapters are not that important and can be skipped. Focus on classes, modules, basic language things. learn about pip (possibly from google) learn how to write requirements.txt and setup.py for your own programs use pycharm for program development, configure it so it uses python 3.7.4 when executing the python programs do it in a terminal not from within pycharm (see Section [¿sec:python-editors?](#)).

We recommend that you install pycharm and use it. We have simple videos in the python section that showcases this.

You do not have to do some of the more advanced python concepts. Focus initially on the language and learn how to do classes as this will be extremely helpful.

7.1.3 Week 3: Sep 6 - Sep 13

7.1.3.1 Cloud Architectures

This week we will focus on architectural definitions of cloud computing. We like that the class engages in a discussion about a very short definition of *cloud computing* and *mainframes* in piazza. We will jointly develop an answer and add it to the handbook at the spots marked with a red circle.

Read the sections:

- Architectures (see Section [¿sec:cloud-architectures?](#))
- NIST Big Data Reference Architecture (see Section [¿sec:nist-bdra?](#))

Students and TA's please complete the red circles, e.g. mostly bibtex references.

7.1.3.2 REST services

Read the REST Service section. This section includes some parts that are not that relevant for this class and we like you to focus on in Section [¿sec:rest?](#) on the sections

- Overview
- OpenAPI REST Services with Swagger (see Section [¿sec:swagger?](#))
- OpenAPI Specification (see Section [¿sec:openapi-spec?](#))
- OpenAPI REST Service via Introspection (see Section [¿sec:openapi-introspection?](#))

You do not in this class need to look at the other Sections about REST, however, if you like to you can. For the project, all projects will be using for REST services the framework used in [sec:openapi-introspection](#).

7.1.4 Week 4: Sep 13 - Sep 20

7.1.4.1 Github as REST service

Read:

- Github REST Services (Section [¿sec:Github-REST?](#))

Recommended:

- Do E.github.issues.6 as this is directly relevant for your projects and can be generalized to other REST services.

Optional:

- Do exercise E.github.issues.5:



The week 4 is under construction and additional items will be added.

7.1.4.2 Practical OpenAPI

Develop an OpenAPI service with not trivial functionality that uses GET, PUT, PATCH and other functions. Make sure to properly secure it. Explore the use of MongoDB as database (MongoDB will be used in your REST services, so it may take some time to master this). We will provide additional material throughout the semester on this. So this task may take multiple weeks and may overlap with your project. Get started early.

7.1.5 Week 5: Sep 20 - Sep 27

In this week you will start exploring virtual machines. For this you will read the section about virtualization (Section **¿sec:virtualization?**). You will be asked to understand what virtualization is and how it differs from typical workstations. You will identify the different levels of virtualization.

You will need to do one of two assignments dependent on which class you are in (naturally you can contribute to both):

All options must be covered by students of the class so please coordinate via github where these tasks will be filed.

Due data is when all sections and chapters are due, but early submissions will be awarded bonus points.

Assignment python virtualenv (Residential students of this class): In addition, we have started the class asking you to use python virtualization while using pyenv. This is not just a recommendation, but also serves as a learning tool in this class. As such we like you to engage in a discussion on Piazza to explain to us what the difference is between virtual machines and python virtual environments. We also like you to explain the difference between pyenv and venv from python version 3. Make sure you do not plagiarize.

Assignment Online Students (Option A): If you have not yet taken any section, you will explore Libvirt on your computer and write a small section on using it practically. Make sure you do not plagiarize.

Assignment Online Students (Option B): If you have not yet taken any section, you will explore Hyper-V on your computer and write a small section on using it practically. Make sure you do not plagiarize.

Assignment Online Students (Option C, for Windows 10 users): If you have not yet taken any section, you will explore Linux Subsystem on your computer and write a small section on using it practically. Make sure you do not plagiarize. Describe how to install python (hopefully pyenv) if there are any differences and cloudmesh cm.

Assignment Online Students (Option D): If you have not yet taken any section, you will explore Hyper-V on your computer and write a small section on using it practically. Make sure you do not plagiarize.

7.1.6 Week 5 and 6: Sep 27 - Oct 4 - Oct 11

7.1.6.1 Sections and Chapters

Online students must select sections and chapters. This is optional for residential students and can be substituted by contributing to the manual page of cloudmesh v4. If a residential student like to do a chapter or section please let us know. However we recently observed that some assignments, (see the previous week) must be done by everyone as we observed that this will increase understanding among all class members.

7.1.6.2 IaaS

Please read (Section `sec:iaas-intro?`). In these weeks you will be exploring IaaS frameworks. As there are many and we like you to program them in python, you will first apply for at least one cloud. One of these clouds must be chameleon cloud as some of our initial implementations can be tested on it easily. In addition to chameleon, you will apply for another cloud account. The free tier from the providers are sufficient.

The clouds can be for example: AWS, Azure, Google, Watson, ...

In addition we recommend that you install and use virtualbox as this simulates a local cloud on your computer.

Thus you have access to 3 infrastructure as a service frameworks

OpenStack, virtualbox, and one more public cloud

We will ask you to run a small program in each of the clouds so that you can compare runtime benchmarks between them. We also ask you to run the same program on your local computer so you have a comparison.

We will announce the program soon but in the meanwhile get your accounts and try out the IaaS services.

7.1.6.3 Libcloud and MongoDB

You will be using libcloud which is introduced in Section [§sec:libcloud-python?](#) for much of your interaction with the IaaS services. Although all of them provide their own python libraries, we like to develop a service mesh and libcloud allows us to provide mostly a uniform interface to virtual machine and other service management.

As we want to store information about the clouds and virtual machines locally, we also like you to study MongoDB which we will be using for the duration of the class including your class projects (see Section [§sec:mongodb-python?](#)).

So the interaction we typically will have

```
IaaS <--> libcloud <--> MongoDB <--> client
```

At a later point in your project you will also do

```
MongoDB <--> REST <--> OpenAPI <--> client
```

For your convenience we have provided a simple mongodb management command in cloudmesh. Please explore it and if you find issues with it improve it. It requires that you fill out the `~/cloudmesh/cloudmesh.yaml` file.

For more information, see also

- <https://cloudmesh-community.github.io/cm/>

All of you will contribute commands and services to cloudmesh to enable a community driven service mesh as part of your project.

As we have provided as part of cloudmesh you already with the basic management functions for MongoDB, as well as a decorator that allows to store any dict into MongoDB very easily.

We would like you to explore this functionality

Your assignment is to first generate a `cmds` command with your first name. This assignment is mandatory for all students

Then generate some dict, that you will return as part of a function that you decorate with the database decorator `@DatabaseUpdate`

The data must have values for `cloud`, `kind` which are used to identify the collection as well as ideally a `name` for each entry in the collection.

For more information see Section `¿sec:mongodb-python?` and Section `¿sec:mongodb-cloudmesh?`.

7.1.7 Week 7: Oct 11 - Oct 18

7.1.7.1 MapReduce

This week you will start learning the basics of MapReduce concept and its use-cases to understand what MapReduce is. The MapReduce section (see Section `¿sec:map-reduce?`) gives an introduction and describes the basics of the MapReduce concept.

Watch the video lecture that describes the software echo system around MapReduce which is listed in Section `¿sec:map-reduce?`.

Understand the Wordcount example discussed in the, word count example can

be thought of as a “Hello world” example into MapReduce.

7.1.8 Fall Break: Oct 18 - Oct 20

We recommend that you spend your time wisely and if you are behind evaluate if this break can be used to catch up.

7.1.9 Week 8: Oct 20 - Oct 25

7.1.9.1 Hadoop and Spark This week you will further look into MapReduce

through Apache Hadoop, Hadoop is one of the earliest open source implementations of the MapReduce concept. Even though Hadoop has now been replaced by faster frameworks such as Spark, Flink and Twister2. It is still important to understand the basic concepts around Apache Hadoop.

Read and understand the Hadoop echo system as described in Section **sec:hadoop-introduction?**. Even though Hadoop is rarely used as a MapReduce framework, other parts of the echo system such as HDFS(Hadoop File System) and Yarn are still very popular.

Watch and understand the pagerank example which is a popular and important algorithm which can be solved cleanly with MapReduce

Install Hadoop and try out some of the examples to get a better understanding about how the framework operates.

Read the Spark section to understand how spark was developed to improve upon Apache Hadoop and how it achieves better performance for Iterative MapReduce applications

MANDATORY: Read the Scientific Writing with markdown ePub, this is required before you make and contributions to the book to make sure that you understand the format of the book and the correct notations. The ePub is available in [Scientific Writing](#)

MANDATORY Project Milestone: Review of your cloudmesh commandline commands specified in docopt. Please make sure you have your command either

specified in the cm directory or in your hid. make sure to add it to your README, so i can find it. The command ust be available Next week Friday Mar 8, 9am.

7.1.10 Week 9: Oct 25 - Nov 1

7.1.10.1 Containers

Container technology is an important topic in cloud computing and has been gaining more and more traction and demand over the past few years. This week you will learn container technology and the concepts and tools that you would need to master. see Section **¿sec:container-intro?** to get a introduction into container technology and its motivations

Docker is one of the most important tools in container technology, Section **¿sec:docker-intro?** provides an introduction into docker technology and internal details of docker.

Read the installation instructions that are provided to install docker locally on your machines to test docker commands

Section **¿sec:docker-file?** describes how docker image files are constructed. Create your own docker images and run it to grasp the concepts

Read about docker hub and its capabilities, check how to push and pull images as needed from docker hub

Docker swam allows user to manage large amounts of docker containers, read about docker swarm in Section **¿sec:docker-swarm?**

Kubernetes is another widely used container management framework. See Section **¿sec:kub-intro?** to read about Kubernetes in more detail

MANDATORY Project Milestone: OpenAPI specification, We like to review your OpenAPI specification that is motivated by your cloud mesh command. Have your openAPI specification ready Mar 15, 9am.

MANDATORY: comprehension assignment VM management, All students in

class must be showcasing for one cloud that they can start, stop, and login into a VM in a cloud. This has to be demonstrated through a script or python program. The program must read the credentials from a configuration file. The script must demonstrate that a command can be run on your vm with ssh. Starting and stopping the VM via the GUI does not count. This must be done by Mar 15

7.1.11 Week 10: Docker Cluster Nov 1 - Nov 8

See Section [¿sec:docker-cluster?](#) for an introduction to terminology and mechanisms for managing multiple containers across multiple hosts. Get hands on experience with with docker clusters by creating a Docker container with Hadoop to explore the Map/Reduce framework. See Section [¿sec:hadoop-docker?](#) for MapReduce examples.

7.1.12 Week 11: Kubernetes Nov 8 = Nov 15

See Section [¿sec:kub-intro?](#) for an introduction to Kubernetes and install `minikube` for development on a local machine. This `minikube` allows users to play with virtual machines managed by either virtual box drivers and docker containers. Learn those terminologies and setup/shutdown virtual machines with Kubernetes.

7.1.13 Week 12: Kubernetes Nov 15 - Nov 22

See Section [¿sec:kub-fs?](#) for using Kubernetes on futuresystems. It introduces you on how to use the Kubernetes cluster on FutureSystems using multiple nodes. Learn to build a service in a cluster environment.

7.1.14 Thangsgiving break Nov 24 - Dec 1

We recommend that you spend your time wisely and if you are behind evaluate if this break can be used to catch up. This is also a good time to continue to work on your project.

7.1.15 Project Due Date Dec 1

Your project will be reviewed and based on feedback you need to improve it. Last day to submit improvements is Dec 13.

7.1.16 Week 13: Go or Julia Dec 1 - Dec 6

This week allows you to experiment with either Go or Julia. Please, select one and try to develop a RES service in either one of the languages.

Go: See Section [¿sec:go-intro?](#) and Section [¿sec:go-language?](#) for the introduction to the Go programming language. Install Go development environment from the section Section [¿sec:go-installation?](#). Learn the concurrent programming and message chain communication in Go. Learn to develop the rest service with OpenAPI in Go from the sections: Section [¿sec:go-openapi?](#) and Section [¿sec:go-links?](#).

Julia: TBD

7.1.17 Week 14: Dec 6 - Dec 13

Improve the project

7.1.18 Week 15: Dec 13 - 20

Project improvements accepted but may not be reflected in the project. We encourage you to finish the project Dec 1. Review of the project is typically one week. Submission after Dec 13. may result in an incomplete.

8 QUICKSTART

8.1 OVERVIEW

8.1.1 Requirements

We recommend that you know one programming language using object oriented programming. Although most activities are done in Python, this programming language does not have to be python as it can easily be learned throughout the semester. No background in cloud computing is needed.

8.1.2 Time Commitment

Any class at an university requires a significant time commitment. Due to the different background the students have it is difficult to predict the actual time needed. On average we see that students spend 6 hours on the class if they do participate on a weekly basis. Students with little programming experience spend up to 12 hours.

The biggest issue we see is that students may devote all their time to others classes and not our Labs or assignments as they are more free form than typical rigid homework assignment. However remember they are still graded and need to be handed in on time.

8.1.3 Course Material List

Course material will be distributed as ePubs. However you will be required to research also some topics on the internet. Cloud Computing is such an evolving field that changes frequently and often the internet is the best resource. To allow you to learn the newest things, we ask you to help us updating the ePub and to use additional resources as appropriate.

The list includes:

- Introduction to Python for cCloud Computing
- Introduction to Linux for Cloud Computing

- Plagiarism Certificate from IU



- [Cloud Technologies, Gregor von Laszewski](#)
- <https://github.com/cloudmesh-community/proceedings-fa18/tree/master/project-report>

A single page has about 1000 words in ACM format. References are managed in bibtex as documented in:

- <https://github.com/cloudmesh-community/book/blob/master/vonLaszewski-writing-markdown.epub?raw=true>

8.1.4 Help

If you take our class, please use piazza to ask for help. This is important as questions may be answered by different TAs based on expertise. Please, do not send e-mail to the instructors. TAs are not allowed to answer e-mail send to them personally.

8.1.5 How to Take this Class

This class is attended by students with greatly different backgrounds and time schedules. To be most flexible and adress all students there are two different ways on how you can take this class.

- *Way 1: Free form.* The preferred way for most students. Here you simply look at the Syllabus table for the semester and identify whatever section you feel like reading. However, make sure you follow our weekly **Lab activities** and readings.
- *Way 2: Linear form.* The lecture notes are just like a book in the syllabus table. We add sections in a logical fashion to the book once they become available. The section form a linear progression and you can go through the book in linear fashion.

Please note that we have set aside a recommended set of weekly *Lab activities*. These are integrated into your final grade, so it is important to do them. They also will let us know if you have any issues and if you participate in this class. As such your activity will be integrated in a participation grade. You certainly can work ahead. Lab activities are simply graded with pass fail.

Typically, Lab activities are supposed to be completed within one week as it alerts us of problems you might have that we can then address.

Lab activities will not receive any credit if you are a residential student and the activity has not been completed within one week. However, residential students will get two **Delay a Lab for One Week** passes that you can apply to any of the Labs and still get credit.

If you are an online student we recommend that you finish the Labs also in the same week. However, you will get eight **Delay a Lab for One Week** passes that you can apply to any of the Labs.

Please note that if you would need to postpone a lab for two weeks, you need to use two passes. Lab passes expire one month before the last day of class. You will have to complete all labs by that time. No credit will be given if this deadline is missed for any delayed Labs as TAs must focus their attention on project support.

8.1.6 Assignments

Besides the Lab's, we have only three main assignments in this class. The Lab's will prepare you towards achieving these assignments. Students that skip the Lab's will miss some important aspects that make their projects easier to execute.

The first two assignments are created in such a way that they could (but do not have to) support your class project. The three assignments are:

- Section contributions
- Chapter contributions
- Project

The assignments will be determined on individual basis within the first 3 weeks while making sure there is no overlap with other students or teams. This will help avoiding plagiarism.

As part of the class, we expect you to get familiar with topics related to cloud computing beyond what we have written in the lecture notes. This is done in Sections, Chapters with Examples. The assignments related to them are done so you do not have to do tests or exams as part of this class. They showcase your understanding of the field making tests and exams unnecessary.

8.1.6.1 Section

A section is a small document (usually about 200 words but no less) that explains a topic which is not yet in the handbook or improves an existing section significantly. It is typically multi-paragraphs long and can even include an example if needed. Sections can be theoretical, or explain a programming related aspects including an example. Typically an A student contributes 2 such sections.

Sample sections contributed by students include:

- Section Microsoft Nafik Data Center in the Datacenter Chapter
- Section Lambda Expressions in *Introduction to Python*


8.1.6.2 Chapter with Example

A chapter is a much longer topic and is a coherent description of a topic related to cloud computing. A chapter could either be a review of a topic or a detailed technical contribution. The minimal length of a chapter is about 800-1000 words. However you will see that it is much easier to write more than less. You will be contributing a unique and **significant** chapter that can be used by other students in the class and introduces the reader to a general topic related to the topic of the class.

Chapters can be theoretical, but most often students prefer the creation of practical contributions. When doing a practical section, it is expected to develop a practical example demonstrating how to use a technology. The chapter and the

practical example can be done together. We do not like to use the term tutorial in our writeup *at all*. We will not accept screenshots for simple command line examples or programs. A sample of a student contributed chapter is

- GraphQL

 *It is expected from you that you self identify a section or a chapter as this shows competence in the area of cloud computing. If however you do not know what to select, you must attend an online hour with us in which we identify sections and chapters with you. The emphasis here is that we do not decide them for you, but we identify them **with** you. Technologies that are not repeatable due to enormous cost or licensing issues need to get prior approval. Naturally, the technology you write about needs to be related to cloud computing.*

8.1.6.3 Project

The objective of the project is to define a clear problem statement and create a framework to address that problem as it relates to cloud computing.

A project is the major activity that you chose as part of your class. This includes a project report* or *manual* and working project code. You will create a significant non-trivial project related to cloud computing and cloud engineering. Up to three people can collaborate. The project could be built on top of a previous project but must have significant additions or modifications. If a previous project is used, a detailed discussion is to be held on what has been improved.

In this class it is especially important to address the reproducibility of the deployment. A test and benchmark, possibly including a *downloadable* dataset, must be used to verify the correctness of your approach.

8.1.6.3.1 License

All projects are developed under an open source license, such as the Apache 2.0 License. You will be required to add a LICENCE.txt file and describe how other software, if used, can be reused in your project. If your project uses different

licenses, please add a README.md file that describes which packages are used and what licenses these packages have.

8.1.6.3.2 Project Report

A project report is to be delivered and continuously improved throughout the semester in github. It includes not just the analysis of a topic, but a short description of the Architecture and code, with **benchmarks** and demonstrated use. Obviously it is longer than a term paper and includes descriptions about reproducibility of the application. A README.md is provided that describes how others can reproduce your project and run it. Remember that tables and figures do not count towards the paper length. The following minimal length is required:

- 800 words, one student in the project
- 1200 words, two students in the project
- 1600 words, three students in the project

The report is written in markdown and checked into github. A Report could be substituted by a manual and benchmarks.

For certain projects, the requirement of a report can be waved or is reduced while replacing it with more programming activities. This includes

- Any project that enhances cloudmesh
- Building a large cloud cluster with Rasperry Pi's
- Any Application project showcasing NIST big data reference architecture use (there is a hard deadline of the NIST project by Dec 1st).

However you still have to do a manual and usage examples, benchmarks and pytests for them.

8.1.6.3.3 Project Code

This is the **documented** and **reproducible** code and scripts that allows a TA to replicate the project. In case you use images, they must be created from **scratch locally** and may not be uploaded to services such as dockerhub. You can, however, reuse approved vendor uploaded images such as from ubuntu or centos

or other linux distributions. All code, scripts, and documentation must be uploaded to github.com under the class specific github directory. Reproducibility must be shown with pytests.

8.1.6.3.4 Project Data

Data is to be hosted on IU's Google drive, if needed. If you have larger data, it should be downloaded from the internet. It is your responsibility to develop a download program. The data **must** not be stored in GitHub. You are expected to write a python program that downloads the data either from the Web or IU's data storage.

8.1.6.3.5 Work Breakdown

This is an appendix to the document that describes in bullet form who did what in the project. This section comes on a new page after the references. It does not count towards the page length of the document. It also includes explicit URLs to the git history that documents the statistics to demonstrate more than one student has worked on the project. If you can not provide such a statistic or all check-ins have been made by a single student, the project has shown that they have not properly used git. Thus, points will be deducted from the project. Furthermore, if we detect that a student has not contributed to a project, we may invite the student to give a detailed oral presentation of the project.

8.1.6.3.6 Bibliography

All bibliography has to be provided in a bibtex file that **MUST** either be validated with **jabref** or with **emacs**. There is **NO EXCEPTION** to this rule. Please be advised doing references right takes some time so you want to do this early and throughout the semester. What would take less than 5 minutes a week, could quickly add up to multiple hours at the end of the semester. Please note that exports of Endnote or other bibliography management tools do not lead to properly formatted bibtex files, despite their claims of doing so. You will have to clean them up and we recommend to do it the other way around. Hence, the easiest way to manage your bibliography is with jabref or emacs. Make sure **labels** only include characters from [a-zA-Z0-9-]. Use dashes and not underscore and colons (_ , :) in the label. We will deduct points if you submit an invalid

bibtex file to github. So please make sure your file is validated. You can even create your own checks with tools such as biber.

8.1.6.4 Project Deliverables

In general, any project must be deployable by the TA. If it takes hours to deploy your project, please talk to us before final submission. This should not be the case. Also, if it takes 100 steps, we are sure you can automate them ... as you are likely doing something wrong or have not thought about cloud computing where we tend to automate most of the steps.

You have plenty of time to execute a wonderful project but you need to work consistently on it. Starting one week before the deadline will not work.

In general your deliverables will include the following (We will address and explain them in a Lab):



- Provide benchmarks.
- Take results in a cloud services and your local PC (ex: Chameleon Cloud, echo kubernetes). Make sure your system can be created and deployed based on your documentation.
- Each team member must provide a benchmark on their computer and a cloud IaaS, where the cloud is different from each team member.
- We require you to write one or more pytest's that deploys, run, kill, view, clean that deploys your environment, runs application, kills it, views the result and cleans up afterwards.
- For python use a requirements.txt file and develop a `setup.py` so your code can be installed with `pip install .`
- For docker use a Dockerfile
- (If not exempt) write a report that typically includes the following sections:
 - Abstract

- Introduction
 - Design
 - Architecture
 - Implementation
 - Technologies Used
 - Results
 - Deployment Benchmarks
 - Application Benchmarks
 - (Limitations)
 - Conclusion
 - (Work Breakdown)
- Your paper will **not** have a *Future Work* section as this implies that you will do work in future and your paper is incomplete. Hence we would not grade it. Instead, you can use an optional “Limitations” section.
 - Do communicate your status and add a *Workbreakdown* section in which you outline which tasks need to be done and by whom in case of a group project. Once you have done a task simply include maker a task as follows

* [done, Gregor] This was gregors task to showcase how to mark it

8.1.7 Submission of sections and chapters and projects

Sections and subsections are to be added to the `book` github repo. Do a pull request. Initially they will be managed in your own github repo that we will set up for you. They will be added to the book after they have been reviewed and approved.

The headline of the section needs to be marked with a  if you are working on it and marked with a  if you want it to be graded and have included all hids for people that contributed to that section.

In addition, simply add them to your README.yml file in your github repo. Add the following to it (I am using a18*516-18 as example).

Please look at <https://github.com/cloudmesh-community/fa18-516-18> and <https://raw.githubusercontent.com/cloudmesh-community/fa18-523->

[62/master/README.yml](#) for an examples. Please note that if you work in a group, the code and report is supposed to be only stored in the first hid mentioned in the group field. If you store it in multiple directories your project will be rejected.

```
section:
  - title: title of the section 1
    url: https://github.com/cloudmesh-community/book/chapters/...
  - title: title of the section 2
    url: https://github.com/cloudmesh-community/book/chapters/...
  - title: title of the section 3
    url: https://github.com/cloudmesh-community/book/chapters/...
chapter:
  - title: title of the chapter
    url: https://github.com/cloudmesh-community/fa18-516-18/blob/master/chapter/whatever.md
group: fa18flys-523-62 fa18-523-69
keyword: whatever
project:
  - title: title of the project
    url: url in your hid space or that of your partner
    group: fa18-523-62 fa18-523-69
    keyword: kubernetes, NIST, Database
    code: the url to the code
other:
  - activity: spell checked md document
    url: put url here
```

You **MUST** run `yamllint` on the `README.yml` file. YAML errors will cause point deductions. Any invalid yaml file will result in point deductions. Please keep your yaml file valid at any time. Our scripts depend on it. The yaml file will also be used to create a list for TAs to review your deliverables. If it's not in the yaml file it will not be reviewed. Please note that it is not sufficient to just run `yamllint`, but to compare your yaml file carefully with the `README.yml` examples. Make sure you do the indentation with 2 spaces, do not use the TAB character and make sure you use the list and attribute organization with proper dash placement. Work with the TAs if you have difficulties. If you copy, only copy from the raw content in github..

8.1.8 Participation

In addition to these artifacts, there will also be a participation component in class that will be determined based on your productive contributions to piazza to help others that have questions and contributions to the books to, for example, improve sections with spelling, grammer or content. We can see from the github history if you conducted such improvements. Make sure that technical contributions, work on all OSes and are not just targeting a single OS if the improvement is of general nature (exceptions apply).

8.2 ASSIGNMENTS

We have three graded assignments all other activities are geared towards supporting these assignments. The first two assignments are created in such a way that they could (but do not have to or may not) support your class project. The assignments are posted in piazza in the `assignments` folder. The three assignments are:

- Section contributions
- Chapter contributions
- Project

To conduct these assignments you will need a number of accounts.

8.2.1 Account Creation

As setting up your computer is not really an assignment but a precondition, we do not grade you on this. Furthermore, we want you to post a formal Bio to piazza after you have gained access. This serves two purposes. First, it tells us we can communicate with you within piazza, and second, you can introduce yourself to others in piazza to potentially build project or study teams.

As part of the class you will need a number of accounts

- piazza.com (used for communication)
- github.com (used for project and other class artifacts)
- futuresystems.org (free docker account)
- chameleoncloud.org (free cloud account)
- google.com (optional)
- aws.com (optional)
- azure.com (optional)
- Watson from IBM (optional)
- google IaaS (optional)

A survey is to be filled out in the first week of class. It includes your github.com account that we need to create your github directory in which you will submit your open source project.

8.2.2 Sections, Chapters with Examples

As part of the class, we expect you to get familiar with topics related to cloud computing beyond what we have written in the lecture notes. This is done in Sections, Chapters with Examples. These assignments are done so you do not have to do other weekly homework or tests and exams. They showcase your understanding of the field.

Section:

A section is a small section that explains a topic that is not yet in the handbook or improves an existing section significantly. It is typically multi-paragraphs long and can even include an example if needed. Sections can be theoretical, or programming related sections. Typically an A+ student contributes more than 5 such sections or replaces multiple sections with an additional or longer chapter.

Sample sections contributed by students include:

- Section [Microsoft Nafik Data Center](#)
- Section [Lambda Expressions](#)


Chapter with Example:



A chapter is a much longer topic and is a coherent description of a topic related to cloud computing. A chapter could either be a review of a topic or a detailed technical contribution. Several Sections (10+) may be a substitute for a chapter. You will be contributing a unique **significant** chapter that can be used by other students in the class and introduces the reader to a general topic related to the topic of the class. You will create a *unique non existing* chapter that can be shared with other students of this class. Chapters can be theoretical, but most often students prefer the creation of practical contributions. When doing a practical section, it is expected to develop a practical example demonstrating how to use a technology. The chapter and the practical example can be done together. We do not like to use the term tutorial in our writeup *at all* . Chapters that focus on theory may not have an example and as such it can be substituted by a longer text. In case no example is provided the example can be substituted in some cases by a

review or comparison.

In case the chapter includes an example (or multiple) it is recorded in a reproducible fashion. We will not accept screenshots for simple command line examples as documented in our [Section Notation](#) To remind you examples augment Chapters as well as Sections. A sample of a student contributed chapter is

- [GraphQL](#).

 *It is expected from you that you self identify a section or a chapter as this shows competence in the area of cloud computing. If however you do not know what to select, you must attend an online hour with us in which we identify sections and chapters with you. The emphasis here is that we do not decide them for you, but we identify them **with** you. Technologies that are not repeatable due to enormous cost or licensing issues need to get prior approval. Naturally, the technology you write about needs to be related to cloud computing.*

Sample Topics that could form a section or chapter are clearly marked with a  or a . and are integrated in the github issues, see [Section Gitissues](#). There will be plenty in the handbook, but you are welcome to define your own contributions. Discuss them with us in the online hours. To guarantee that they are unique and others know about it you will file a github issue for it once it is approved by us via a discussion either in an online hour or piazza.

In addition you can use the following volume to identify technologies that may interest you.



Cloud Technologies, Gregor von Laszewski

- <https://github.com/cloudmesh/technologies/blob/master/vonLaszewski-cloud-technologies.epub?raw=true>

8.2.3 Mini Projects that could Substitute a Chapter

In some special cases it is possible to substitute the chapter and/or section contributions with an additional mini project, that however still has to be documented. An example of such a mini project is our `cm-burn` command that creates Raspberry PI OS based on manipulation of the file system

- <https://github.com/cloudmesh/cm-burn>

Please note that this is not less work than developing a chapter and/or sections. You still will have to do a class project as the mini project does not substitute the class project.

A mini project may be related to topics such as:

- Raspberry PI
- Sechi Disk Partial Image Analysis
- NIST OpenAPI definition and implementation

Mini projects must be suggested and approved by Dr. Gregor von Laszewski.

If you choose the *Cloudmesh Version 4* project, you are allowed to substitute Sections and Mini Chapters with a single large project that requires less documentation. However, you must contribute to this project in a significant way. The project link is

- <https://github.com/cloudmesh-community/cm>

and contains an alpha version which is currently worked on. Please do not expect that the documentation is up to date. Residential students most frequently chose this as option. All programming in the project is done in python and weekly updates are discussed in meetings or in github.

8.2.4 Project

Project

A project is the major activity that you chose as part of your class. The default case is an implementation project that requires a *project report* and project code. You will create a significant non-trivial project related to cloud computing and cloud engineering. Up to three people can collaborate.

The project could be built on top of a previous project but must have significant additions or modifications. If a previous project is used, a detailed discussion is to be held on what has been improved. We will discuss projects in one of our future lectures. There are a lot of examples in our other class publications.

License:

All projects are developed under an open source license, such as the Apache 2.0 License. You will be required to add a LICENCE.txt file and describe how other software, if used, can be reused in your project. If your project uses different licenses, please add a README.md file that describes which packages are used and what licenses these packages have.

Project Report:

A project report is to be delivered and continuously improved throughout the semester in github accessible to the instructors. It includes not just the analysis of a topic, but a short description of the Architecture and code, with **benchmarks** and demonstrated use. Obviously it is longer than a term paper and includes descriptions about reproducibility of the application. A README.md is provided that describes how others can reproduce your project and run it. Remember that tables and figures do not count towards the paper length. The following length is required:

- 6 pages (ACM format), one student in the project
- 8 pages (ACM format), two students in the project
- 10 pages (ACM format), three students in the project

Please note that we no longer use the ACM format but, instead, use markdown. All project reports are to be delivered in markdown. A sample format is located in github at

- <https://github.com/cloudmesh-community/proceedings-fa18/tree/master/project-report>

A single page has about 1000 words in ACM format. References are managed in bibtex as documented in:

- <https://github.com/cloudmesh-community/book/blob/master/vonLaszewski-writing-markdown.epub?raw=true>

Project Code:

This is the **documented** and **reproducible** code and scripts that allows a TA to replicate the project. In case you use images, they must be created from **scratch locally** and may not be uploaded to services such as dockerhub. You can, however, reuse approved vendor uploaded images such as from ubuntu or centos. All code, scripts, and documentation must be uploaded to github.com under the class specific github directory.

Data:

Data is to be hosted on IU's Google drive, if needed. If you have larger data, it should be downloaded from the internet. It is your responsibility to develop a download program. The data **must** not be stored in GitHub. You are expected to write a python program that downloads the data either from the Web or IU's data storage.

Work Breakdown:

This is an appendix to the document that describes in bullet form who did what in the project. This section comes on a new page after the references. It does not count towards the page length of the document. It also includes explicit URLs to the git history that documents the statistics to demonstrate more than one student has worked on the project. If you can not provide such a statistic or all check-ins have been made by a single student, the project has shown that they have not properly used git. Thus, points will be deducted from the project. Furthermore, if we detect that a student has not contributed to a project, we may invite the student to give a detailed oral presentation of the project.

Bibliography:

All bibliography has to be provided in a bibtex file that **MUST** either be validated with **jabref** or with **emacs**. There is **NO EXCEPTION** to this rule. Please be advised doing references right takes some time so you want

to do this early and throughout the semester. What would take less than 5 minutes a week, could quickly add up to multiple hours at the end of the semester. Please note that exports of Endnote or other bibliography management tools do not lead to properly formatted bibtex files, despite their claims of doing so. You will have to clean them up and we recommend to do it the other way around. Hence, the easiest way to manage your bibliography is with jabref or emacs. Make sure **labels** only include characters from [a-zA-Z0-9-]. Use dashes and not underscore and colons (_ , :) in the label. We will deduct points if you submit an invalid bibtex file to github. So please make sure your file is validated. You can even create your own checks with tools such as biber.

8.2.4.1 Project Deliverables

The objective of the project is to define a clear problem statement and create a framework to address that problem as it relates to cloud computing. In this class it is especially important to address the reproducibility of the deployment. A test and benchmark, possibly including a dataset, must be used to verify the correctness of your approach. Projects related to NIST and Cloudmesh Version 4 focus on the specification and implementation. The report here can be smaller but the contribution must be includable in the specification document.

In general, any project must be deployable by the TA. If it takes hours to deploy your project, please talk to us before final submission. This should not be the case. Also, if it takes 100 steps, we are sure you can automate them ... as you are likely doing something wrong or have not thought about cloud computing where we tend to automate most of the steps.

You have plenty of time to execute a wonderful project but you need to work consistently on it. Starting one week before the deadline will not work.

The deliverables included need to be updated according to your specific project throughout the semester, for example if you do Edge Computing some deliverables will be different. In general your deliverables will include the following:

- Provide benchmarks.

- Take results in a cloud services and your local PC (ex: Chameleon Cloud, echo kubernetes). Make sure your system can be created and deployed based on your documentation.
- Each team member must provide a benchmark on their computer and a cloud IaaS, where the cloud is different from each team member.
- We strongly suggest to create a Makefile with the tags `deploy`, `run`, `kill`, `view`, `clean` that deploys your environment, runs application, kills it, views the result and cleans up after wards. You are allowed to have different makefiles for the different clouds and different directories. Keep the code and directory structure clean and document how to reproduce your results.
- For python use a requirements.txt file and develop a `setup.py` so your code can be installed with `pip install .`
- For docker use a Dockerfile
- Write a report that typically includes the following sections:
 - Abstract
 - Introduction
 - Design
 - Architecture
 - Implementation
 - Technologies Used
 - Results
 - Deployment Benchmarks
 - Application Benchmarks
 - (Limitations)
 - Conclusion
 - (Work Breakdown)
- Your paper will **not** have a *Future Work* section as this implies that you will do work in future and your paper is incomplete. Hence we would not grade it. Instead, you can use an optional “Limitations” section.
- Do communicate your status add a *Workbreakdown* section in which you

outline which tasks need to be done and by whom in case of a group project. Once you have done a task simply include maker a task as follows

* [done, Gregor] This was gregors task to showcase how to mark it

8.2.5 Project: Virtual Cluster

As previously stated, all students can contribute and participate on the creation of the Cloud service mesh project called Cloudmesh Version 4 that we will be using throughout the class to improve and interface with cloud and container frameworks. This project is managed at:

- <https://github.com/cloudmesh-community/cm>

Residential students will work on this project and meet in a discussion group to work on it while having weekly assigned tasks they define with us. Online students are welcome to join this project and their tasks will be discussed in online hours. You will also need weekly time to work on it. Online students that are in the Bloomington area, are welcome to join our residential meetings (Please contact Gregor via piazza).



Work on cloudmesh cm4, cm-burn or the NIST REST Specification document can reduce the project report deliverable (in size) while substituting it with a **significant** larger programming contribution.

- In case of cm4 we may write collaboratively a report but focus on the manual first
- In case of cm-burn we will work on a report that we publish in the Raspberry community
- In case of NIST you will be expected to contribute to the NIST Specification.

In addition to interfacing with clouds via an API, we are also interested in displaying the interactions in Javascript. So if you have Javascript skills this may be a good opportunity for you to contribute to the project with your previous knowledge.

8.2.6 Submission of sections and chapters and projects

Sections and subsections are to be added to the `book` github repo. Do a pull request. Initially they will be managed in your own github repo that we will set up for you. They will be added to the book after they have been reviewed and approved.

The headline of the section needs to be marked with a  if you are working on it and marked with a  if you want it to be graded and have included all hids for people that contributed to that section.

In addition, simply add them to your README.yml file in your github repo. Add the following to it (I am using a18-516-18 as example).

Please look at <https://github.com/cloudmesh-community/fa18-516-18> and <https://raw.githubusercontent.com/cloudmesh-community/fa18-523-62/master/README.yml> for an examples. Please note that if you work in a group, the code and report is supposed to be only stored in the first hid mentioned in the group field. If you store it in multiple directories your project will be rejected.

```
section:
- title: title of the section 1
  url: https://github.com/cloudmesh-community/book/chapters/...
- title: title of the section 2
  url: https://github.com/cloudmesh-community/book/chapters/...
- title: title of the section 3
  url: https://github.com/cloudmesh-community/book/chapters/...
chapter:
- title: title of the chapter
  url: https://github.com/cloudmesh-community/fa18-516-18/blob/master/chapter/whatever.md
```

group: fa18flys-523-62 fa18-523-69 keyword: whatever project: - title: title of the project url: url in your hid space or that of your partner group: fa18-523-62 fa18-523-69 keyword: kubernetes, NIST, Database code: the url to the code other: - activity: spell checked md document url: put url here

You **MUST** run yamllint on the README.yml file. YAML errors will cause point deductions. Any invalid yaml file will result in point deductions. Please keep your yaml file valid at any time. Our scripts depend on it. The yaml file will also be used to create a list for TAs to review your deliverables. If it's not in the yaml file it will not be reviewed. Please note that it is not sufficient to just run yamllint, but to compare your yaml file carefully with the README.yml examples. Make sure you do the indentation with 2 spaces, do not use the TAB

character and make sure you use the list and attribute organization with proper dash placement.

More details will be posted throughout the semester. Work with the TAs if you have difficulties, but it's good to copy and paste from a working example. Depending on class, your format may be slightly different. We will inform you if additional format changes are needed throughout the semester.

8.2.7 Participation

In addition to these artifacts, there will also be a participation component in class that will be determined based on your productive contributions to piazza to help others that have questions and contributions to the books to, for example, improve sections with spelling, grammar or content. We can see from the github history if you conducted such improvements. Make sure that technical contributions, work on all Oses and are not just targeting a single OS if the improvement is of general nature (exceptions apply).

8.3 COURSE SYLLABUS TABLES

Please note that the the dates may change throughout the class. It is your responsibility to check weekly and download the lecture notes on a weekly basis to see the updates.

- All assignments are due **2 weeks** before semester end.

8.3.1 Proposed Lecture Timeline

The times indicate when to start a particular technology documentation or lecture. It may take you some weeks to complete some of the sections. If you know the topic well, it may take you less time so you should move ahead. For example while it may take some time for some to learn python, others have the knowledge already and should move on. The book can be used as reference material in that case.

Legend markings

- Class released +
- Class under development - or **0**

Week	Unit	Titl	e Desc	ription
+	1	1	Introduction	Gregor von Laszewski
+	1			Class summary
+	1			Definition of Cloud Computing
+	1	2	Tools	Tools and Services
+	1			- Virtual Box
+	1			- Vagrant
+	1			- Github
+	1			- Linux
+	2	3	Python	Python
+	2			- Introduction
+	2			- Installation
+	2			- Interactive Python
+	2			- Editors
+	2			- Basic Language Features
+	2			- Modules
+	2			- Data Management
+	2			- Matplotlib
+	2			- Cloudmesh Commandshell CMD5
+	2			- OpenCV
+	2			- Secchi Disk
+	3			Data Center
+	3	4	Architectures	- NIST Big Data Reference Architecture
+	3			- Cloud Architectures
+	3			REST

+	3			- OpenAPI and Swagger
+	3			- OpenAPI Specification
+	3			- OpenAPI Service
+	3			- Github as Rest Service
+	4	5	Virtualization	Virtualization, Qemu, KVM, Virtual machines
+	4	5	Virtualization I	- Qemu
+	4	6	Infrastructure	Infrastructure as a Service
+	4			- Azure
+	4			- AWS
+	4			- OpenStack
+	5		Chameleon Cloud	- Chameleon Cloud
+	5			- Resources
+	5			- Hardware
+	5			- Charge
+	5			- Quick start
+	5			- KVM user guide
+	5			- CLI
+	5			- Horizon
+	5			- Heat
+	5			- Baremetal
+	5			- FAQ
+	5	8	Programming	Python for Cloud Computing,
+	5			- Libcloud
+	6		Virtualization II	Containers, Docker, Kubernetes
+	7	9	Map/Reduce	Map/Reduce , [Hadoop] (#sec:hadoop-introduction), Spark
+	8	10	Messaging	Messaging

+	8	11	Messaging	- MQTT
+	8			- GraphQL
+	13		Go	Go Introduction
+	13			- Go Links
+	13			- Go Install
+	13			- Go Editors
+	13			- Go Language
+	13			- Go Libraries
+	13			- Go cmd
+	13			- Go Cloud
+	13			- Go REST
+	13			- Go for the Cloud

8.3.2 Assignments Timeline

Students must conduct all assignments listed here. They must conduct one project of type A, B, C, or E. The project is selected in the first 3 weeks of the semester and conducted throughout the rest of the semester.

	Week	Unit	Title	Description
+	1	A0	Survey	Fill out the Survey before Friday in the first week
+	2	A0	Bio	Post your formal Bio into Piazza
+	0	A1	Sections	Contribute significant sections. Do not develop redundant or duplicated content.
+	0	A2	Chapter	Contribute a significant chapter that may use your section to the class documentation. Do not develop redundant or duplicated content.
+	0	A3	Project Report draft due	Develop a draft for the project. This is a hard deadline as we integrate your draft into a proceedings over the

berak.

+	0	A3	Project Type A	Build a cloud cluster out of Raspberry PIs
+	13		Project Type B	Build a Significant OpenAPI REST Service
+	13		Project Type C	Contribute to the new Cloudmesh code
+	13		Project Type D	Your own Project Type A, B, C, D (upon approval)

- The project is a long term assignment (and are ideally worked on weekly by residential students). It is the major part of the course grade.

(*) Sections and chapters prepare you for documenting a technical aspect related to cloud computing. It is a preparation for a document that explains how to execute your project in a reproducible manner to others.

- all times are in EST

Additional lectures will be added that allow easy management of the project. These lectures can be taken any time when needed.

Date	Unit	Title	Description
+ anytime	1	Scientific Writing with markdown	
+ anytime		Plagiarism	How to avoid plagiarism and cheating
+ anytime		Markdown	How to use markdown
+ anytime	1	Scientific Writing II	
+ anytime	1	only relevant for the bibtex section fro this class, we will not use LaTeX	

How to write

+
anytime

Writing a Project Report

a high quality
Project report
following our
template

+
anytime

Bibliography Management

How to easily
manage
bibliographies
for your
Project
Report

8.3.3 Group Breakdown Checkpoint

Please note that all checkins in case of group projects are visible in github. If we detect that a group member has disproportionally fewer contributions to the project than other project team members we will invite the team for a special section in which each team member needs to explain what has been done. This is to avoid that a team member unfairly relies on other team members and does not contribute to the project. For example, a project containing three members should contribute the work of three team members and not fewer. This also means that if you want to work in a project you need to vet your team members. Do choose them based on capability and make sure they are a good fit for your project.

9 OVERVIEW

9.1 ORGANIZATION

This class is an online class. Online classes require you to be very disciplined in order to execute the tasks necessary for the class in time. It is your responsibility to organize the lessons so that you can complete them not only by the end of the semester, but also in time for conducting your assignments. This is a great opportunity for you to structure the class based on your availability. The classes are attended by two different set of students. One set are remote online students, while to other are residential students. For the residential students we have a mandatory in person meeting that takes place at the posted location and hours once a week. For pure online students we have weekly online hours that we will identify based on our availability and a doodle poll.

[Figure 65](#) showcases the different parts of the class. If you have taken a previous class with Gregor von Laszewski you are able to continue your previous project upon approval. It must however be a significant improvement.

There will not be any bonus projects or tasks to improve grades. Instead make sure your deliverables of the few assignments are truly outstanding.

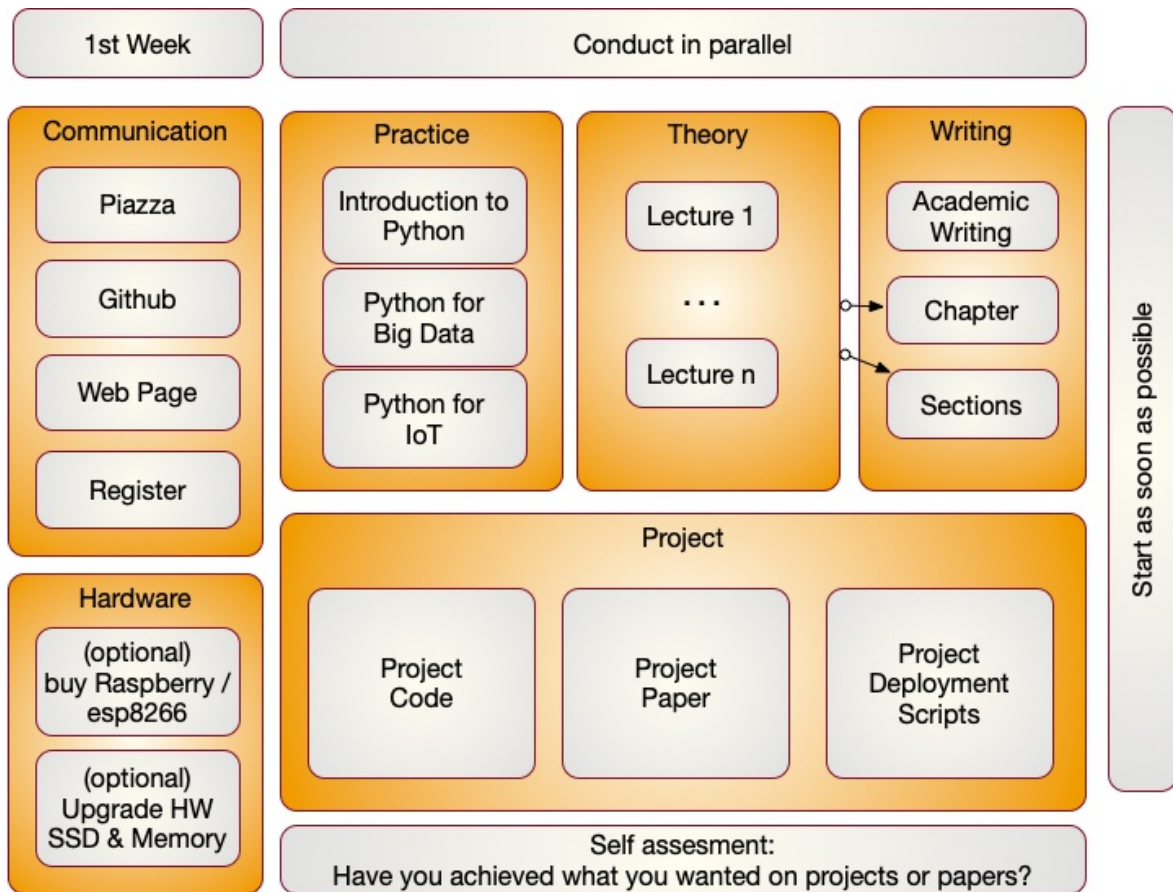


Figure 65: Components of the Class e516, e534 and e616

The content for this class will be available through a series of documents that will be regularly updated and are linked from this document. All communication is done with Piazza.

9.1.1 First Week

In the first week we will be introducing you how we communicate to you. Naturally you need to register for the class. Once you register you need to set up a number of services.

9.1.2 IoT Hardware

As part of this class you also have the ability to take part in some Internet of Things related projects but also a small cloud cluster based on Raspberry PI's. Residential students will have access to a 5 node raspberry pi cluster. Online students could by at least 3 PI's but that is optional. You can do the class without

the PI educational component. However it is fun to build your own cloud with them.

9.1.3 Access to Clouds

As part of the course you will also need access to a computer. We will try our best to provide you with access to suitable computers for the class, but do be reminded that the amount of time and access to supercomputers and clouds we offer is limited. Our class policy is to use the compute resources only when you really need them. Thus you **must** shut down your VMs when they are not in use. It would be a violation of class policy if we would find out through an analysis of the cloud logs that you unnecessarily keep your VMs running. Thus we will implement a **strict policy** that you must record yourself how many hours you run VM's and provide this information to us. We will then compare that time with the time recorded by the computer system as well as with your target application and will deduct points from your project if you can not justify why you have not shut down your VMs. A resource section needs to be added to your report justifying the used resources.

Why is this such a big deal you may ask? For example we estimate if every student in class violates this policy it would cost about \$200000 to rent the time for this on a public cloud. Due to this high cost, we no longer tolerate deliberate violations of the policy and will terminate your account. Furthermore, violators will have to find alternative resources to conduct their projects while not using our resources. In our case the problem is even beyond the issue of cost as our allocation on the clouds would be terminated due to abuse and **no student**, including those that follow policies, could use the cloud. It may take weeks to reestablish cloud access and would effect every student in class.

We will provide clarification for accessing cloud resources and teach you how to avoid getting in such a situation. I am sure that a future employer of yours will be real happy if you have a deep understanding of resource vs. cost estimate.

Listing the used computer time for your project is part of your report.

9.1.4 Using Your Own Computer

In many cases however you could and are recommended to use your own

personal computer, but make sure the computer is up-to-date. We also like to make sure that you do not use a work computer as you need to avoid that when you develop a cloud program you do not by accident introduce a security risk on your machine. This does not mean that you need to buy a new computer, or need to upgrade it. However, if you consider an upgrade of an older machine please consider the following.

These days we recommend that your computer has a solid-state drive and fast memory (put as much memory in your machine as is supported). We recommend 16 GB of main memory which gives you enough space to run containers, virtual machines and naturally the main operating system. We found that students with only 8GB could do the work but it was slow. In some cases the memory to conduct their projects was not sufficient. Make sure you follow your upgrade guide to your computer and by suitable memory chips. In most cases you have to buy them in **pairs** and make sure all chips in your computer are the same. When it comes to buying a solid-state drive, make sure that you buy one that is compatible with motherboards bus speed. As you may want to reuse your solid-state drive at a later time I suggest to get a 6GB/s SSD and not a 3GB/s.

In case of Windows, you could also get yourself a USB stick or external SSD drive and place ubuntu on it. You could then use your bios to boot of from that drive. This way you do not have to modify anything on your computer. This method works very well for most computers and allows you to use the maximum memory while for example using ubuntu.

Students that only had a chromebook and took this class gave us the feedback that they are too inconvenient as they do not allow you to program directly in python on them and the ssh terminals to login to other computer although working are not supporting the GUI tools.

Another option is (if money is an issue) you can buy a Raspberry Pi and edit your programs there and when satisfied run them on a cloud.

We also like to remind you that this course does not require you to purchase expensive text books, thus the money you save on this could be used in upgrading your hardware or renting yourself from your own money time on AWS. However, be careful with the cloud its easy to spend lots of money there if you are not careful.

9.1.4.1 Self Discipline

As this class has no graded tests and only few graded homework, we like that you deliver an **exceptional** project report. Instead of focusing on preparing for tests we provide you with the opportunity to **explore** without the pressure of grades. However you should not give up or take the easy way out or it will effect you in your project execution. Also, to achieve your best do not just say: *We do not have a test, so let me not do this weeks assignment, let me do it next week.* After a couple of times with this attitude you will be in big trouble. All this requires discipline. For example, if you believe you are so good that you can do a project within one week before deadline, you will **certainly fail**. To avoid this and to introduce discipline, you will also be monitored on progress and we check your github for activities which will be part of the participation grade.

It will be up to you to assess what you want to deliver before handing it in to us. Self assessment or a check with other students is a real good way to do that. You should not expect to get an A if you yourself are not convinced about your project or are unsure about it. Common sense prevails.

9.1.4.2 Fun

I hope you have fun and are able to integrate in the projects your own thoughts and interests.

We have quotes form students such as

“This is the best class I have taken ...”

or

“I really enjoyed taking this class and having maximum flexibility to schedule the lectures. ...”

or

“The lessons learned from this class were adopted within my company. ...”

or

“I wanted to sincerely thank you for all the guidance you provided in this course. My learning in cloud computing has enhanced a lot because of this course and also because of your continuous guidance. ...”

or

“Thanks to the material you taught I got a job at Intel. ...”

Furthermore, you should know that the way we teach the class has also been adopted in STEM classes. As a result a team coached by Gregor von Laszewski won an award at the FLL Robotics World Championship. The certainly had lots of fun and integrated their own ideas into the project that won the award. This year the team has also won the best presentation award in Indiana State, showcasing that presenting your results is an important aspect that others recognize early on.

9.1.4.3 Uniqueness

We will try to have every project or paper to be non overlapping with another topic, If there are overlaps we may ask you to modify your focus.

9.1.4.4 Continuation

If you like to put additional effort in the project, the report could be made to a conference or workshop paper. Dr. von Laszewski is happy to help as co-author.

9.1.5 Parallel Tracks

In this class we have three parallel tracks.

9.1.5.1 Track 1: Practice

Track 1 introduces you to using python for Big Data. We recommend that you do know a programming language for any of our courses. Learning a

programming language is not part of the hours you spend for this class. It is an additional time requirement that you must plan for. Maybe you want to take for example a python programming language class at the same time. This can also be done in self study. Although you do not need to know any programming language, it is certainly useful as it will make this course much easier for you. We had students that had no prior programming knowledge and successfully completed the course. So we know it can be done. The course is designed in such a fashion, that there is enough time to learn programming and do a project.

We provide you with a general introduction to Python. This includes enough knowledge so you can conduct a project with it. We will reinforce this knowledge while exposing you optionally to IoT devices that you can program in Python such as the Raspberry PI. Residential students that have access to 100 Raspberry PIs. They can then compare the compute power of that cluster with your own Laptop, or a cluster hosted in the cloud.

We will build on these technologies to introduce you to python libraries that can be used for big data.

Optionally, we also offer you the chance to integrate DevOps into your projects which is typically covered in I524, e516, e534, and e616 for However, we have a real simple solution while using our own cloudmesh cmd5 to provide an easy interface to reproducible environments that could be used by anyone in the class.

9.1.5.2 Track 2: Theory

The theory track includes a number of online lectures that introduces you to a variety of topics related to Big Data. You have especially the opportunity to become part of a project that would contribute to the understanding and the development of a Big Data Architecture developed in collaboration with NIST. Other topics that are covered include naturally Cloud computing.

9.1.5.3 Track 3: Writing

This track will introduce you into how to write a sections, a chapter, and examples while conducting proper bibliography management. Knowing how to write is a preparation for your term project.

You will be writing a section of substantial length and difficulty using markdown. We suggest to do the section in a team as this allows one to write about it and another person to test it. We like to avoid that all students take the same topic. We will use github to avoid that everyone chooses same topic. Knowing how to write a section will be a valuable exercise as the TAs will also need a section in which you describe how to execute your project. It also prepares you in a gentle form on writing your project report. In general you need to look at previous examples for the class to identify project reports.

9.1.5.4 Track 4: Project

The major deliverable of the course is a term project or paper. The exact details will be posted on the Web page in this document. The important part is that you start on this project once you are sufficiently familiar with Track 1-3. However you can also use the project to for example learn python and engage in a goal oriented learning activity while working towards implementing your project and integrating the python lessons that you encounter. The same is valid for the theory.

It is **expected** that you identify a suitable performance benchmark for the project and that you learn how to apply this analysis as well as justify it. It is part of the learning outcome that you determine this instead of us giving you a topic.

9.1.6 Plagiarism

In the first week(s) of class you will need to read the information about plagiarism. If there are any questions about plagiarism we require you to take a course offered from the IU educational department.

Warning:

If we find cheating or plagiarism, your assignment will be receiving an F. This especially includes copying text without proper attribution. We are required to follow IU policy and report your case to the dean of students who may elect to expel you from the university. Please understand that it is your doing and the instructors have no choice as to follow university policies. Thus, please do not blame the

instructors for your actions. Excuses such as “I missed the lecture on plagiarism”, “I forgot to include the original reference as I ran out of time”, “I did not understand what plagiarism is” do not apply as we explicitly make the policies clear. This applies to all material prepared for class including assignments, exercises, code, sections, tutorials, papers, and projects. If there is no time, do not submit and instead of an F ask for an incomplete. In fact if you know you have plagiarized, do not even have us review your paper.

For more information on this topic please see:

- <https://studentaffairs.indiana.edu/student-conduct/misconduct-charges/academic-misconduct.shtml>

Furthermore you are supposed to review our lecture material on plagiarism and take the plagiarism test. The information is located at:

- [Scientific Writing with Markdown](#)

In Piazza a form will be posted that will ask you for your passing ID. If the form is not yet posted, please be patient till it is.

9.2 RESEARCH INTERESTS



Learning Objectives

- Identify interesting topics you can do research in with Dr. von Laszewski.
 - Decide if you also like to do an independent study in addition to the class with him.
 - For online students that have a job he recommends not to take more than 6 credit hours per week.
 - For full time residential students (graduates) he recommends not to take more than 9 credit hours per week.
-

In this lecture we are presenting some of the research interests of Gregor von

Laszewski, the instructor of this class.

This Includes:

- Cloud Computing
- Cloud and Big Data Computing Architectures
- Cloud and Edge Computing
- Scientific Impact Analysis

Dr. von Laszewski also conducts independent studies with students. All work he is involved in is done as open source projects.

It is recommended that you watch the following introduction video:



[Research Interest Dr. Gregor von Laszewski \(14:13\)](#)

9.3 E516: ENGINEERING CLOUD COMPUTING

Students considering to take this class are recommended to take a look at:

- <https://github.com/cloudmesh-community/book/blob/master/syllabus/e516.pdf>

Sample chapters can be reached if you download the PDF document locally and click on the links.



Learning Objectives

- This is the syllabus of the class. It will be updated throughout the semester, so look here for changes.
 - Identify if this is the right class for you.
 - Enroll if you want to take this class.
 - Avoid an incomplete.
-

- Lecture Notes: <https://github.com/cloudmesh-community/book/blob/master/vonLaszewski-cloud.epub>
- Piazza: <https://piazza.com/iu/fall2019/516>
- Indiana University
- Spring 2019
- Faculty: Dr. Gregor von Laszewski (laszewski@gmail.com)
- Credits: 3
- First online class released: Jan. 07, 2019
- Residential students Discussion: 11:15A-12:15P Fridays I2 150, bring your laptops
- Prerequisite(s): Knowledge of a programming language, the ability to pick up other programming languages as needed, willingness to enhance your knowledge from online resources and additional literature. You will need access to a “modern” computer that allows using virtual machines and/or containers. If such a system is not available to you can also use cloud vms we provide and if you opt to do so one or more Raspberry’s PI. All residential students will have access to a total of 200 Raspberry PIs. Online students can opt to purchase one or more based on our materials list that we will release throughout the semester. All students will have access to a cloud.
- This page is maintained and updated at [e516: Engineering Cloud Computing](#)
- Course Description URL: <https://github.com/cloudmesh-community/book/blob/master/chapters/class/e516-engineering-cloud-computing.md>
- [Registrar information and Other related classes](#)

This is an introductory class. In case you like to do research and more advanced topics, consider taking an independent study with Dr. von Laszewski.

9.3.1 Course Description

This course covers basic concepts on programming models and tools of cloud computing to support data intensive science applications. Students will get to know the latest research topics of cloud platforms, parallel algorithms, storage and high level language for proficiency with a complex ecosystem of tools that span many disciplines.

9.3.2 Course Objectives

The course has the following objectives:

- Provide a basic introduction to cloud computing
- Introduce the concept of cloud data centers
- Get familiar with cloud infrastructure as a Service such as OpenStack, Azure, or AWS
- Get familiar with cloud infrastructure such as Docker and Kubernetes
- Program cloud services
- Understand the differences between virtual machines and containers
- Develop sophisticated programming language independent REST services
- Learn advanced programming models for clouds such as Map/Reduce, Messaging, and GraphQL
- Exploration of Go for cloud computing
- Demonstrate knowledge of clouds while developing a significant project
- Explore state-of-the-art cloud technologies and services while providing a section and summary and commenting on its use for the cloud
- Learn how edge computing is enhancing cloud services and infrastructure
- Learn how to set up a cloud based on using commodity hardware

9.3.3 Learning Outcomes

- Be able to explain the concepts of the cloud computing paradigm including its paradigm shift, its characteristics, and the advantages. Contrast them with the challenges and disadvantages.
- Be able to identify infrastructure and programming models needed to support real world applications.
- Be able to implement a real world application or deploy a cloud and its services.
- Be able to conduct sophisticated performance analysis of cloud services.
- Be able to communicate the results through tutorials, manual, and reports.
- Be able to work in a team to develop collaboratively software or contribute collaboratively to develop sections explaining how to use clouds.

9.4 SYLLABUS

For the syllabus table please see the the syllabus table

9.4.1 Assessment

This course is focusing on the principal “Learning by Doing” which is assessed by simple graded and non-graded activities. The assessment may include comprehension of the material taught, programming assignments, participation in online discussion forums, or the contribution of additional material to the class showcasing your comprehension.

The comprehension is also measured by the development of a tutorial in markdown that can be distributed and replicated to other students. This is done in preparation for the project that must include a simple deployment and runtime instruction set.

The main deliverable of the class is a project. The project is assessed through the following artifacts:

1. Deployment and install instructions,
2. Project report (typically 2-3 pages per month, tutorial and chapters can be reused if possible),
3. Working project code that can be installed and executed in reproducible manner by a third party
4. Code developed by the project team distributed in github.com
5. Project progress notes checked into github throughout the semester. Each week the project progress is reported and will be integrated into the final grade.
6. three discussions or progress reports with the instructors about your project

The grade distribution is as follows

- 10% Comprehension Activities
- 10% Sections
- 10% Chapter
- 70% Project

As the project is the main deliverable of the course it is obvious that those starting a week before the deadline will not succeed in this class. The project

will take a significant amount of time and fosters the principal of “Learning by Doing” at all stages throughout the semester.

The class will not have a regular midterm, but it is expected that you have worked on your project and can provide a snapshot of the progress outlining the goals of the project and how you will achieve these goals till the end of the semester.

The final Project is due Dec. 1st. Issues with your project ought to have been discussed before this deadline with the TA’s The TAs will in the next 14 days go over the projects and evaluate mayor and minor issues that you may be able to fix without penalty. Larger changes will receive a grade penalty. The last fix (upon approval) possible will be Apr 21st.

9.4.1.1 Incomplete

Please see the university regulations for getting an incomplete. However, as this class uses state-of-the-art technology that changes frequently, you must expect that an incomplete may result in significant additional work on your behalf as your project may need significant updates on infrastructure, technology, or even programming models used. It is best to complete the course within one semester.

9.5 COURSE POLICIES

We describe briefly some class policies.

9.5.1 Discussion via Piazza

1. All communication is done in Piazza. It is an IU approved communication tool and superior to CANVAS discussion list
2. CANVAS is only used with students that are not in Piazza. The only messages they will get is to activate Piazza and use the class Piazza
3. You are allowed to use whatever calendar system you like.
4. We will not use CANVAS calendar, however you can manage that yourself. CANVASS allows you to add events such as assignment deadlines.
5. Piazza is FERPA compliant <https://piazza.com/legal/ferpa>

9.5.2 Managing Your Own Calendar

From time to time we get the question from a very small number of students why we are not using or uploading the assignment deadlines and the assignment descriptions to CANVAS. The reason for this is manifold. First, our class has different deadlines for different students within the same class. This is not supported by CANVAS and if we would use CANVAS leads to confusion and clearly shows the limitation of CANVAS. Second, we are teaching cloud computing. CANVAS is not a tool that you likely will use after graduating. Thus we are providing you the ability to explore industry standard tools such as github to maintaining your own tasks and deadlines, while for example using github issues (see the section about github). We highly recommend that you explore this as part of this class and you will see that managing the assignments in github is **superior** to CANVAS. Naturally you can not make that assessment if you are not trying it. Thus we like you to do so and it is part of any assignment in your class to use github issues to manage your assignments for this class.

However, if you still want to manage your tasks in CANVAS, you can do so. CANVAS allows you to create custom events, so if you see an assignment in piazza or the handbook, you are more than welcome to add that task yourself to your own CANVAS tasks. As we have only a very small number of assignments this will not pose a problem either for graduate or undergraduate. Being able to organize your deadlines and assignment with industry accepted tools is part of your general learning experience at IU.

Obviously, this makes it also possible to use any other task or calendar system that you may use such as google calendar, jira, microsoft project, and others.

As you can see through this strategy we provide the most flexible system for any student of the class, while giving each student the ability to chose the system they prefer for managing their assignment deadlines. It is obvious that this strategy is superior to CANVAS as it is much more general.

9.5.3 Online and Office Hours

To support you we have established an open policy of sharing information not only as part of the class material, but also as part of how we conduct support. We

establish the following principals:

- in case of doubt how to communicate address this early in class and attend online hours;
- all office hours if not of personal nature are open office hours meaning that any student in class can be joined by other students of the class and all meeting times are posted publicly. This includes in person office hours with TAs. Other students are allowed to listen in and participate.
- it is in your responsibility to attend in person classes and online hours as we found that those that do get better grades. For residential students participation in the residential classes may be mandatory. International students may need to check university policies.
- instructors of this class will attempt within reason to find suitable times for you to attend an online hour in case you are an online student.

9.5.3.1 Office Hour Calendar

Online Students:

- Online hours are prioritized for online students, residential students should attend the residential meetings.

Residential Students:

- Residential students participate in the official meeting times. If additional times are required, they have to be done by appointment. Office hours will be announced publicly. All technical office hours are public and can be attended by any student. Online hours are not an excuse not to come to the residential class.

However Residential students can in addition to the residential class use the online student meeting times. However, in that case online students will be served first. It is probably good to check into the zoom meeting and identify if the TA has time. They will be in zoom.

Meeting times and phone numbers are posted in your piazza in the Resources section

9.5.4 Class Material

As the class material will evolve during the semester it is obvious that some content will be improved and material will be added. This benefits everyone. To stay up to date, please, revisit this document on weekly basis. This is practice in any class.

9.5.5 HID

You will be assigned an hid (Homework IDentifier) which allows us to easily communicate with you and does allow us to not use your university ID to communicate with you.

You will receive the HID within the first couple of weeks of the semester by the TA's.

9.5.6 Class Directory

You will get a class directory on github.com and not the iu github. For that reason you will be asked to give us a github id so we can create a openly accessible directory for you in which you can collaborate with the students of this class. The directories are only used to store the artifacts of the class. As all artifacts are supposed to be open source github.com provides us with the service that millions of professionals and researchers use for their work.

9.5.7 Notebook

All students are required to maintain a *class notebook* in github in which they summarize their weekly activities for this course in bullet form. This includes a self maintained list of which lecture material they viewed and what they worked on in each week of the class.

The notebook is maintained in the class github.com in your hid project folder. It is a file called notebook.md that uses markdown as format. Notebooks are expected to be set up as soon as the git repository was created.

You will be responsible to set up and maintain the `notebook.md` and update it accordingly. We suggest that you prepare sections such as: Logistic, Theory, Practice, Writing and put in bullet form what you have done into these sections during the week. We can see from the github logs when you changed the `notebook.md` file to monitor progress. The management of the notebook will be part of your discussion grade.

The format of the notebook is very simple markdown format and must follow these rules:

- use headings with the # character and have a space after the # Use `# Week X: mm/dd/yyyy - mm/dd/yyyy` as the subject line for each week
- use bullets in each topic.
- Do not refer to section numbers from the ePub in your notebook as they can change. Instead use the section name or headline and possibly a URL. When using URLs in md format they must be enclosed in `<>` or `[text](URL)`

Please examine carefully the sample note book is available at:

- <https://github.com/cloudmesh-community/hid-sample/blob/master/notebook.md>

The `notebook.md` is not a blog and should only contain a summary of what you have done.

9.5.8 Blog

You can maintain your own optional blog. However, the blog will not be used for grading. Do not include sensitive information in either the blog or the notebook. A blog is not a replacement for the notebook. If something does not go so well, do not focus on the negative things, but focus on how that experience can be overcome and how you turn it to a positive experience. Be positive in general.

9.5.9 Waitlist

The waitlist contains students that are unable to enroll in a section of a course. Students choose to add themselves to the waitlist. They are not automatically added, but choose to do so intentionally based on the status of the course. There are two reasons for students to be on the waitlist. The first, and primary, reason is that the class is already at the scheduled, maximum capacity. Since there are no seats available, the student can elect to add themselves to the waitlist. The second reason is that the students' own schedule has a time conflict. This occurs when they are trying to enroll in a class that overlaps with the time of a class they are already enrolled in.

Students are moved from the waitlist to the regular section during a daily batch process, and not in real time. The process is not in realtime because the registrar receives many requests to increase capacity, decrease capacity, and change rooms. If the process were real time there would be a catastrophe of conflicts.

Students are moved from the waitlist in chronological order that they added themselves to the waitlist. If you are still on the waitlist there are no spaces free, the batch process has not run for the day, or the student in question has a schedule conflict.

Faculty are not able to selectively choose students from the waitlist.

How long does the waitlist process stay active?: The automated processing of the waitlist ends on Thursday of the first week of class. At this time the waitlist will no longer be processed. As the residential class starts on Friday, this may cause issues. Either talk to the department on Thursday or show up on Friday. Most likely there will be spaces left. Students on the waitlist at that time will remain on the waitlist, but remain there until the student decides to change their registration. Students may not do that, because they get assessed a change schedule fee.

Students tell me they still want to enroll after the first week of classes. How do they do this?

Beginning Monday, after the first week of class students begin to use the eAdd process to do a late addition of the course. The request is routed to the professor of record on an eDoc and the faculty will be notified via email. Faculty can deny or approve based on whatever criteria they wish to apply. If the faculty member

approves, the eDoc is electronically forwarded to the Academic Operations office and we will approve the late add **if the room capacity** allows the addition, otherwise we must deny the addition because of fire marshal regulations. Many times, there are seats in a classroom/discussion/lab, but because other students have not *officially* dropped, enrollment is still at capacity.

After everything, a student that was unable to enroll in the class attended all year and completed all course work as if they had enrolled. Can the student get credit and can I give the student a grade?

Yes. There is a provision for a late registration - contact our office if this occurs. Students will be assessed a tuition fee at the time of late or retroactive registration.

9.5.10 Registration

The Executive Associate Dean for Academic Affairs requires starting Spring 2018 that students that are not officially enrolled, can not register at the end of the class if they in-officially took the class. Please make sure that within the first month you have enrolled. If we do not see in CANVAS, you are not in the class. In case you are on a waitlist it is your responsibility to work with the administration after the waitlist is over to be added to the class by getting permission from the School.

9.5.11 Auditing the class

We no longer allow students to audit the class because:

- Seating in the lecture room is limited and we want foster students that enroll full time first.
- The best way to take the class is to conduct a project. As this can not be achieved without taking the class full time and as auditing the class does not provide the full value of the class, e.g. not more than 10% of the class. Hence, we do not think it is useful to audit the class.
- Accounts and services have to be set up and require considerable resources that are not accessible to students that audit the class.

9.5.12 Resource restrictions

- It is not allowed to use our services we offer as part of the class for profit (e.g. just enrolling in the class to use our clouds).
- In case of abuse of available compute time on our clouds the student is aware that we will terminate the computer account on our clouds and the student may have to conduct the project on a public cloud or his own computer under own cost. There will be no guarantee that cloud services we offer will be available after the semester is over. Projects can be conducted as part of the class that do not require access to the cloud.

9.5.13 Incomplete

Incompletes have to be explicitly requested in piazza through a private mail to *instructors*. All incompletes have to be filed by DATE TO BE ANNOUNCED.

Incomplete's will receive a fractional Grade reduction: A will become A-, A- will become B+, and so forth. There is enough time in the course to complete all assignments without getting an incomplete.

Why do we have such a policy? As we teach state-of-the-art software this software is subject to change, not only within the course, but also after the course. As we may offer some services and only have access to the TA's during the semester it is obvious that we like all class projects and homework assignments to be completed within a semester. Services that were offered during the semester may no longer be available after the semester is over and could adversely effect your planing. It will be in the students responsibility to identify such services and provide alternatives if they become unavailable. We try hard to avoid this but we can not guarantee it.

Furthermore, once an incomplete is requested, you will have 10 month to complete it. We will need 2 month to grade. No grading will be conducted over breaks including the summer. This may effect those that require student loans. Please plan ahead and avoid an incomplete.

The incomplete request needs to be off the following format in piazza:

Subject:
INCOMPLETE REQUEST: HID000: Lastname, Firstname

Body:
Firstname: TBD
Lastname: TBD
HID: TBD
Semester: TBD
Course: TBD
Online: yes/no

URL notebook: TBD
URL assignment1:
URL assignment2: TBD
....
URL paper1: TBD
URL project: TBD

URL other1: TBD

Please make sure that the links are clickable in piazza. Also as classes have different assignments, make sure to include whatever is relevant for that class and add the appropriate artifacts.

In case of an incomplete you may be asked to do additional assignments and assignments that have been adapted based on experience from the class. Please note also that we could reject an assignment if it is identified to no longer reflect the state-of-the-art. All previously submitted assignments such as papers, sections, and so on will be reviewed on this criterion. For example, let us assume you developed a tutorial on technology visit version x. Let us assume that since you completed this task a version $x+1$ comes out. It will be your obligation to update the deliverable. This is also true if the tutorial has been graded previously. The incomplete and the change of the software have at this time negated the originally assigned grade. In most cases the changes may be small. In other cases the changes could be substantial. Hence avoid an incomplete.

Here is the process for how to deal with incompletes at IU are documented:

- <http://registrar.indiana.edu/grades/grade-values/grade-of-incomplete.shtml>

9.5.13.1 Exercises

E.Policy.1

Take the Plagiarism test, See the Scientific Writing I ePub for more details.

9.6 E516 SUMMARY



Learning Objectives

- Obtain an overview of the topics we explore.
 - Identify topics that you are especially interested in.
 - The main deliverable of this class is a project.
 - You **must** dedicate sufficient time on continuous basis for this class.
-
-

Presentation about this Document

-  [Introduction](#)

9.6.1 Introduction

9.6.1.1 Research participation

Research activities by Gregor von Laszewski are directly related to this course allowing you to not only learn about cloudcomputing, but practically participate in ongoing cloud research. Besides the research topics provided, you can also suggest your own as part of a project that you chose. PhD students could benefit from using cloud computing as part of an activity they do plan to do within their PhD. We would expect that you write a paper in collaboration with Dr. von Laszewski and your advisor.

Some short overview about this is provided in [Gregor von Laszewski](#)

In general we are interested in any project that uses the cloud. This can also include some topics that may not be related to Big Data but to enabling services that are just hosted on the cloud or in some cases could be hosted on a cloud but may actually use local clusters or even your Laptop.

Topics of interest include:

- Cloud Computing
- Cloud Computing Architectures
 - NIST Big Data Architecture
 - Federated Cloud Computing
 - Multi Cloud and Hybrid Cloud Architectures
 - HPC in the Cloud
- Cloud and Edge Computing
 - Example: Environmental Autonomous Robot Boat
- Big Data Applications
 - Example: Scientific Impact Analysis
- Monitoring and Visualizing Clouds
- Cloud Portals
- Other Topics

9.6.2 Class communication

We are using for class communication piazza and github. As all projects are supposed to be done as open source projects, we will use github for managing them.

Please see [Class Communication](#) for more details.

9.6.2.1 Topics covered in this class

- Tools and Services
 - Piazza
 - Github
- Definition of Cloud Computing
 - NIST definition

- History of Clouds
- Technologies enabling Clouds
- Service Model
 - IaaS
 - PaaS
 - SaaS
- Data Centers
 - Evolution of the Data Center
 - Today's Data Center
 - Academic Data Centers
- NIST Big Data Reference Architecture
 - Composable Services
 - NIST Big Data Reference Architecture
 - You can contribute!
 - You can benefit!
- Infrastructure
 - Infrastructure as a Service
 - What is infrastructure
 - How to manage infrastructure
 - How to use infrastructure
 - GUI vs commandline vs programming
 - Overview of commercial and educational IaaS
 - OpenStack
 - What is OpenStack?
 - How can you use OpenStack?

- ChameleonCloud
- Azure
 - What is Azure?
 - How can you use Azure?
- AWS
 - What is AWS?
 - How can you use AWS?
- Google
 - What is Google?
 - How can you use Google?
- Watson
 - We may not have time to cover this topic*
 - What is Watson?
 - How can you use Watson?
- Virtualization
 - Introduction to virtualization
 - Qemu
 - KVM
 - Virtual machines
 - Virtual Box
 - Vagrant
 - Containers
 - Introduction to Containers

- Docker
 - Introduction to Docker
 - Dockerfiles
 - Dockerhub
 - Create your own docker images
- Kubernetes
 - Managing containers in kubernetes
- Programming
 - Python for Cloud Computing
 - Python 2 and Python 3
 - Introduction to Python
 - Classes
 - DocOpts
 - CMD, CMD5
 - YAML
 - JSON
 - Module Management
 - setup.py
 - Hosting code on github
 - Installing code from github
 - LibCloud
 - Introduction to libcloud
 - Managing vms
 - REST Services
 - Github as a Cloud Service
 - Accessing Github REST services
 - Mine Github

- Eve
 - Using Eve to develop REST services
 - Integrating swagger docs into Eve
 - OpenAPI
 - Abstracting REST services
 - OpenAPI Specification
 - Generating code from OpenAPI
- Map/Reduce
 - Hadoop
 - Spark
- Messaging
 - MQTT
 - GraphQL
- Go for the Cloud Computing
 - Introduction to Go
 - Developing REST Services with Go
 - Go and Kubernetes
- Julia for Cloud Computing
 - *We may not have time to cover > this topic.*

However this could be chosen by a student as Chapter and section contribution.

- Language motivation
- DocOpts with Julia
- Distributed computing with Julia
- Cloud Computing with Julia
- Reimplementing cloudmesh with Julia

- Accessing MongoDB
- REST services in Julia

- Containers and Julia

- Edge Computing and the Cloud
 - Introduction to PI as Edge device
 - Streaming data from a PI
 - Sensors
 - Autonomous Robot Boat (Can be chosen as Project)

- Other topics

Optional but very fun and useful:

- Maker Lab introduction for residential students.
- Maker Lab certification to operate the laser cutter
- Creating a case for the Raspberry Pi cluster
- Creating laser cut peaces for the Robot boat

We try to set up a 1-2 hour class with certification on Wednesday or Thursday Aug. 22 or Aug. 23, 2018. Certification means you could go to the maker lab by yourself outside of the class.

9.6.3 Assignments

Notebook

You are expected to maintain a notebook in github, that outlines your progress in class. You can either use github, or provide a link to a shared google docs document that you link to the github `notebook.md` page

Bio

You will develop a formal Bio and post it to Piazza.

Chapters

Contribute a significant chapter that may use your section(s) to the class documentation. Do not develop redundant or duplicated content. The chapter can be developed as a team to also allow review by more than one person. However each team member has to develop their own chapter. Plagiarism is not allowed.

Examples:

- Overview of AWS Cloud Services. This chapter provides an overview of AWS Web services.
- Heroku. This chapter provides an overview of Heroku

A chapter must be a reasonable contribution to the class and related to cloud or edge computing.

Section

Contribute a significant section. Do not develop redundant or duplicated content. The section can be developed as a team to also allow review by more than one person. However each team member has to develop their own section. Plagiarism is not allowed.

Example:

- Installation of kubernetes on a Raspberry Pi cluster. This section shows practically how to do the installation
- Heroku. This section provides guidance on how to use Heroku
- Improvement of existing class sections are allowed

A section must be a reasonable contribution to the class and related to cloud or edge computing. Multiple small sections could be a reasonable contribution. Typically sections are correlated within a chapter, you could have however multiple smaller chapters and sections.


Projects

Develop a project in the area of cloud computing. Make sure your project uses a REST services while using OpenAPI as introduced in class. A project report will showcase your comprehension and summarize your

result to others. Alternatively you could use a Project Chapter format that is integrated into the class material. However in this case you need to distinguish your contribution from the regular section and chapter assignment

The project types include

- Project Type A: Build a cloud out of Raspberry PIs while enabling and showcasing an application in nKubernetes, Hadoop, SLURM + OpenAPI Service. It must contain an OpenAPI Rest service.
- Project Type B: Build a Significant OpenAPI REST Service on an existing cloud. As the cloud may already be provisioned your application must be more involved.
- Project Type C: Build an Edge Service Interfacing with a Cloud using OpenAPI Rest services.
- Project Type D: Your own Project upon approval. It must use an OpenAPI REST service.

 *Sections, chapters, and especially Projects are multi-week projects. Do not be tempted to think that other classes are more important and start with your assignments the week before the deadline.*

9.6.4 Scientific Writing

We have made scientific writing extremely simple while using standard tools used by millions of engineers to document their activities. If you follow our templates they are just like forms and you can simply clone the template and fill it out with content you develop. We focus on content and not on the beauty of the text. However as we use templates you will see that the result will be highly professional.

For more information please see our two publications:

- [Scientific Writing I](#), which will introduce you to how to avoid plagiarism and cheating, and use markdown.

- [Scientific Writing II](#), which will introduce you to how to write a high quality Project report following our template and how to easily manage bibliographies for your Project Report

These skills will be extremely useful for many other classes you may take.

9.7 EXAMPLE ARTIFACTS



Learning Objectives

- Identify what other students have done previously.
 - Look at previous chapters, which are collected as technology reviews.
 - Look at previous project reports.
 - Looking at the documents provides you with an initial overview of the scope for the artifacts.
-
-

As part of this class you will be delivering some artifacts that are being graded. Some of them include writing a *chapter* that can be contributed to the class lecture and a project. To showcase you some example artifacts take a closer look at the documents listed in this section. Please also note that you can not duplicate or replicate a students previous work without significant improvements. All material listed here is available online, including all source code.

9.7.1 Technology Summaries

We are maintaining a large list of technologies related to clouds and Big Data at

- <https://github.com/cloudmesh/technologies>

This repository generates the following epub

- <https://github.com/cloudmesh/technologies/blob/master/vonLaszewski-cloud-technologies.epub>

Students that have to contribute as part of their class Technology summaries are

asked to produce meaningful, advertisement free summaries of the technology and indicate in some cases if not obvious show they relate to cloud or big data. The length of such summaries is about 300 words. Students of E516 do not have to contribute to this and will instead focus on programming. Students of I423, I523 and other sections must contribute to it and will get an assignment related to it. We post here the existence of this document also for 516 students. They can voluntarily improve or add sections if they like which will go into their discussion credit.

Please use the following indicators to mark the progress of summaries that you are working on.



ready for review



selected by student so others do not select it and we know what is worked on



needs revision (only assigned by ta, after smiley)

The signs are put as follows. You can view an example at <https://github.com/cloudmesh/technologies/blob/master/chapters/tech/bioconduct>

Ex - Title Of Summary  fa18-xxx-xx

9.7.2 Chapters

Previously we asked students to write a 2 page paper on a topic related to bigdata analytics or cloud technologies (dependent on the course). Example papers are listed bellow

- Use Cases in Big Data Software and Analytics Vol. 1, Gregor von Laszewski, Fall 2017, <http://cyberaide.org/papers/vonLaszewski-i523-v1.pdf>
- Use Cases in Big Data Software and Analytics Vol. 2, Gregor von Laszewski, Fall 2017, <http://cyberaide.org/papers/vonLaszewski-i523->

[v2.pdf](#)

- Big Data Software Vol 1., Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper1/proceedings.pdf>
- Big Data Software Vol 2., Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/paper2/proceedings.pdf>
- Vol 8, Gregor von Laszewski, Spring 2018, <http://cyberaide.org/papers/vonLaszewski-cloud-vol-8.pdf>

This has however resulted in a large number of duplicated material especially in the introductions and motivations. Thus we like this year to have you more focused on the topic and do not write a large introduction on what big data or cloud computing is. Therefore we renamed the 2 paper to a chapter, while you could assume certain things that have already been taught to you and you do not have to repeat it.

9.7.3 Project Reports

The goal of the class is to use open source technology to also write your technical reports. As a beneficial side product, we are able to distribute all previous reports from students to you. In your reports you will be doing a similar report, but will not use the same topic, without a significant improvement from a report already delivered in that area. For big data we have more than 1000 data sets we point to. I am sure you can do a unique project. For engineering cloud there are recently so many new technologies that there is not much chance of an overlap. TA's will review your project proposal, but it is your responsibility to make sure they are unique.

Please note that we do not make any quality assumptions to the published papers. It is up to you to identify outstanding papers.

- Use Cases in Big Data Software and Analytics Vol. 3, Gregor von Laszewski, Fall 2017, <http://cyberaide.org/papers/vonLaszewski-i523-v3.pdf>

- Big Data Projects, Gregor von Laszewski, Spring 2017, <https://github.com/cloudmesh/sp17-i524/blob/master/project/projects.pdf>
- Vol 9, Gregor von Laszewski, Spring 2018, <http://cyberaide.org/papers/vonLaszewski-cloud-vol-9.pdf>

10 GRAPHICS

10.1 GRAPHVIZ

Graphviz is a tool that allows you to visualize structural information with the help of abstract graphs and networks. It is achieved while providing the graph with automatic layout algorithms so you can focus on the creation of dependencies between nodes through edges. The main Web page is located at

- <https://graphviz.gitlab.io/resources/>

10.1.1 Installation

On macOS you can install graphviz with

```
$ brew install graphviz
```

On macOS there is also a graphviz version available that includes a GUI. The link to this software is:

- <http://www.pixelglow.com/graphviz/>

It can be downloaded from

- <http://www.pixelglow.com/downloads/graphviz-1.13-v16.dmg>

There is also an additional tool that is distributed by the community that is called doteditor and can be installed with

```
$ brew cask install doteditor
```

If you have issues with brew cask install, you can also install it by hand while going to

- <https://vincenthee.github.io/DotEditor/>

Online versions of graphviz are also available, but we have not tested them

- <http://www.webgraphviz.com/>
- <https://dreampuf.github.io/GraphvizOnline/>
- <http://viz-js.com/>
- <http://graphviz.it/#/gallery/unix.gv>

There are many more versions available. Please contribute to this section to improve it

10.1.2 Usage

To use graphviz create a dot file and run the following command.

```
$ dot -Tpng filename.dot -o filename.png
```

This will create a png file. Other formats are also possible such as svg, or PDF

```
$ dot -Tsvg filename.dot -o filename.svg  
$ dot -TPDF filename.dot -o filename.pdf
```

For inclusion in latex documents we recommend you create PDF output as it has a much better quality and is smaller in size than png.

10.1.2.1 The Dot Format

An extensive documentation is provided at

- <https://graphviz.gitlab.io/documentation/>

From there we find also the most simplest Hello World Graph>

```
$ echo "digraph G {Hello->World}" | dot -Tpng > hello.png
```

10.1.3 Exercise

Graphviz.1:

Develop a REST service that takes a graph as input and returns a rendered version of the graph in a specified format. Make sure you can pass the format as a parameter.

Graphviz.2:

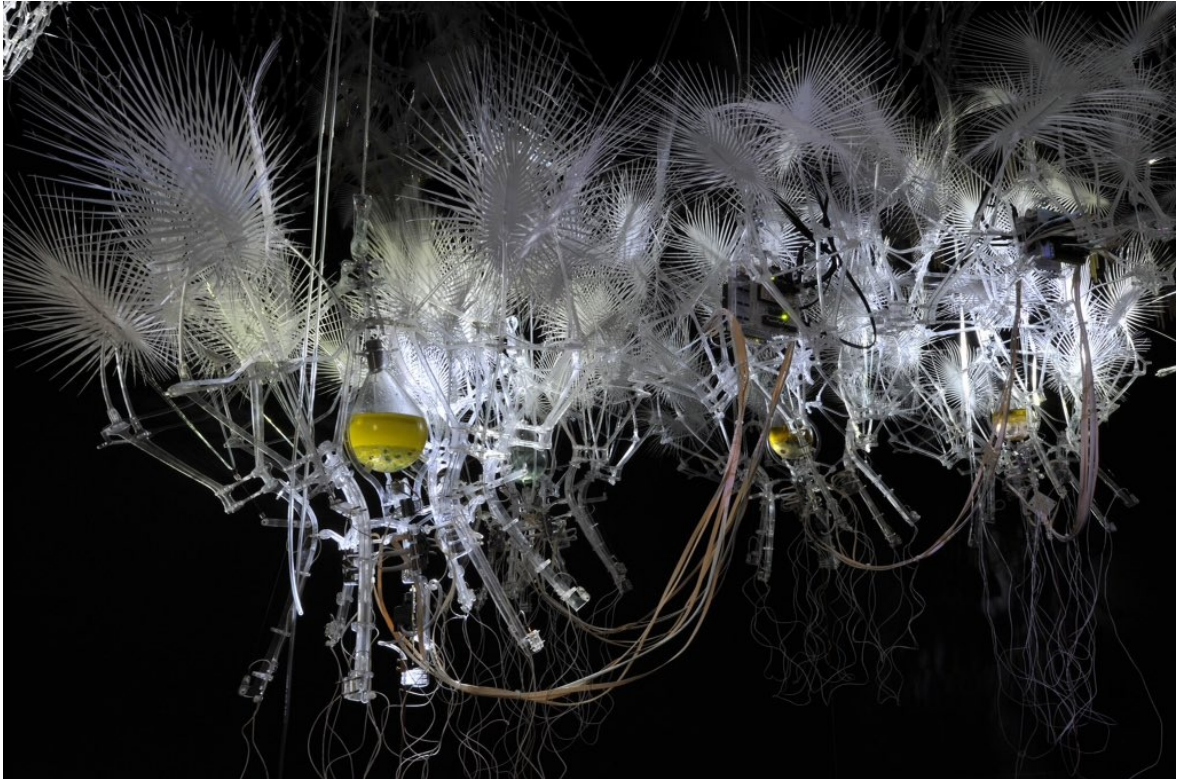
Develop a REST service that takes a graph as input and returns a URL of the rendered graph while storing the output onto a data server. The data server is another rest service, from which the result can be picked up.

Graphviz.3:

For IU students. Develop a REST service that takes a graph as input and returns a URL of the rendered graph while storing the output onto a data server. The data server is another rest service, from which the result can be picked up. Use box and/or google drive that are offered by IU as services. Make sure not to expose your passwords or access keys

11 ART

11.1 SENTIENT ARCHITECTURE



Sentient

Architecture:

Source:

<https://nicolatriscott.files.wordpress.com/2016/03/caaqt-euyam-jpm.jpg>

11.1.1 Introduction

What is it

11.1.2 Existing Deployments

list existing deployments

11.1.3 Impact

why is it important

not only science,

11.1.4 Integration into Cloud Computing and Big Data

how does it fit into cloud computing and big data (Gregor can do that)

11.1.5 Development

S Architecture in practice

11.1.5.1 Snowwhite and the Seven⁺¹ Dwarfs

The ISE department at Indiana University has obtained eight dendrites that were assembled by a number of students so they can be used in class and in research projects. These dendrites can be accessed in Smith Research Center and allow students and faculty members to experiment with them. They are bare dendrites and have no electronics on them. Hence, you will need a hardware device to interact with.

The reason we named them *Snowwhite and the Seven⁺¹ Dwarfs* is based on the fact that the dendrites are white, and they need to be interact with someone. Because of the white color we name the control unit snowwhite. The dendrites that are interacting with it are called dwarfs as this just fits to the name snowwhite. As we actually have 8 and not 7 we added ⁺¹.

We do not recommend to directly attach the wires to boards, as they will draw too much power and destroy the boards. Instead you will need a relay that you control that itself controls the dendrite. These can be:

Arduino:

These boards very simple but provide relative good protection of the output links. The disadvantage is that its interface is in C.

Teensy

Whatever is now on it TBD

Raspberry Pi

We recommend to use a Raspberry Pi as it has a great operating system and is more suited for additional analysis of data within an Edge Computing network than the other two choices. It also allows you to use python which is clearly a plus as most of the material presented here are in Python.



{#fig:snowwhite"}

11.1.5.2 Luddy Hall Installation

architecture drawing

11.1.6 Sentient Cloudmesh

Gregor provides introduction to cloudmesh (probably just pointer to other section)

Gregor provides introduction to cloudmesh.pi

Introduction to parallel programming in python (TBD)

Towards the Cloudmesh Parallel S development environment

11.1.6.0.1 Snowwhite and the Seven⁺¹ Dwarfs

Test environment (8 dendrites)

11.1.6.0.0.2 Luddy Hall

Andreas describes how we program them

(Andreas provides)

than we use cloudmesh to interface with them, maybe we need to just show how we integrate them into mqtt (this is snowwhite) and then we can program them from another pi

Test environment Luddy hall

11.1.7 Alternative Boards

11.1.7.1 Mega 2560

“The MEGA 2560 is designed for more complex projects. With 54 digital I/O pins, 16 analog inputs and a larger space for your sketch it is the recommended board for 3D printers and robotics projects. This gives your projects plenty of room and opportunities.” <https://store.arduino.cc/usa/arduino-mega-2560-rev3> A nice case such as the one offered at Amazon will provide good protection <https://www.amazon.com/Eleduino-Arduino-Mega-2560-Enclosure/dp/B016QE46RQ>

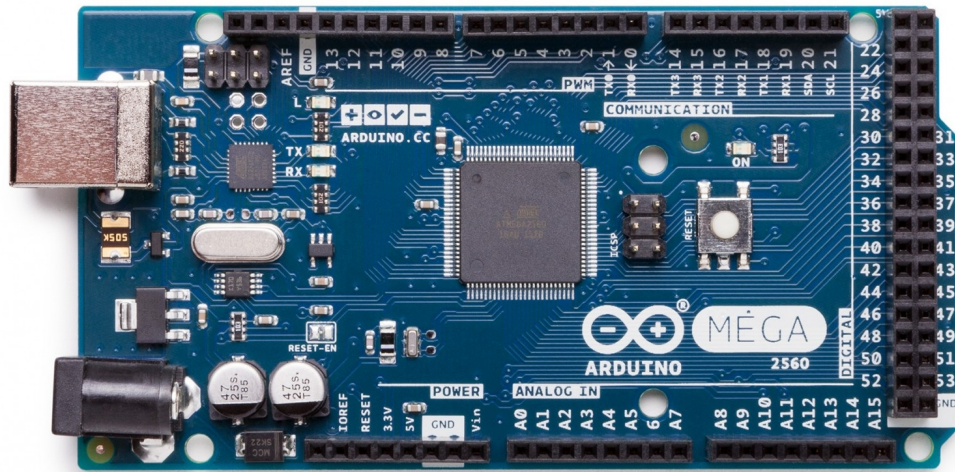


Figure 66: MEGA 2560

11.1.7.2 Programming with Teensy

“The Teensy is a complete USB-based microcontroller development system, in a very small footprint, capable of implementing many types of projects. All programming is done via the USB port.” Source: <https://www.pjrc.com/teensy/>

Current state of programming (whatever they have)

Which version

11.1.8 Exercises

E.Sentient.1

Build dendrite. In this exercise you will be building a dendrite that you can add to the available pool of dendrites.

E.Sentient.2.1

Develop cloudmesh sensor/actuary. In this exercise you will be developing an actuator or sensor interface in object oriented programming methodology. You can see many examples in cloudmesh.py on github.com. You will pick a sensor you have access to and that is not already included in cloudmesh.pi.

E.Sentient.2.2

If you do prefer using another board, the option may exist to develop an interface for the sensor or actuator for this device. If OO programming is not available for that board, a clean design based on functions must be provided. However we believe this is more complex than using the Pi.

E.Sentient.3.1

Develop an mqtt based event publisher and subscription service. Use first LEDs to test your service before you hook up relays and dendrites.

E.Sentient.3.2

Hook up the dendrites to mqtt and control them

E.Sentient.4

Develop sensors that interact with the dendrites

E.Sentient.5

Explore the Page at <https://www.intorobotics.com/alternative-arduino-boards/> that lists a number of PI/Arduino alternative boards provide a non plagiarized table for this chapter and evaluate which could be viable alternatives. If you have one of them we like you to provide a documentation on how to integrate them with the dendrites.

12 OTHER

12.0.1 Fall 2018

The classes are a full term class of 16 weeks. The semester calendar is posted at

<http://registrar.indiana.edu/official-calendar/official-calendar-fall.shtml>

The class begins Aug 20 and Ends Dec 14th. Please do not ask us if you can leave prior to Dec 14th as we do not have the authority to do so. TA's for the class must be available till the grades have been posted to the registrar.

12.0.1.1 Registrar

At time of writing the following classes were listed in the IU Registrar for Fall 2018. The classes may change and new classes may be added. We recommend that you check for updates at:

- <http://registrar.indiana.edu/browser/soc4188/index.shtml>

However, we list some of the information also here so it is easier for you to locate the courses in the different departments which include INFO and ENGR.

E516/E416/CSI-B649 are the same class

This class is an online class but residential students have a meeting time:

- 11:15A-12:15P F I2 150
-

12.0.1.2 ENGR-E 516 Residential

- <http://registrar.indiana.edu/browser/soc4188/ENGR/ENGR-E516.shtml>
- Topic: Engineering Cloud Computing
- von Laszewski, Gregor

- 3 CR
- RSTR ARR ARR ARR
- Class is taught online
- Discussion (DIS), participation in discussion is mandatory and graded
 - 36048 RSTR 11:15A-12:15P F I2 150

12.0.1.3 ENGR-E 516 Online

- <http://registrar.indiana.edu/browser/soc4188/ENGR/ENGR-E516.shtml>
- Topic: Engineering Cloud Computing
- von Laszewski, Gregor
- 3 CR
- 36046 RSTR ARR ARR ARR
- This is a 100% online class taught by IU Bloomington. No on-campus Class meetings are required. A distance education fee may apply; check your campus bursar website for more information. A distance education fee may apply; check your campus bursar website for more information

12.0.1.4 ENGR-E 416

- <https://registrar.indiana.edu/browser/soc4188/ENGR/ENGR-E416.shtml>
- Topic: Engineering Cloud Computing
- von Laszewski, Gregor
- 3 CR
- RSTR ARR ARR ARR
- Class is taught online
- Discussion (DIS), participation in discussion is mandatory and graded
 - 37558 RSTR 11:15A-12:15P F I2 150

12.0.1.5 ENGR-E 416

- <https://registrar.indiana.edu/browser/soc4188/ENGR/ENGR-E416.shtml>
- Topic: Engineering Cloud Computing
- von Laszewski, Gregor
- 3 CR
- RSTR ARR ARR ARR

- Class is taught online
- Discussion (DIS), participation in discussion is mandatory and graded
 - 37558 RSTR 11:15A-12:15P F I2 150

12.0.1.6 CSCI-B 649 Topics in Systems

- <https://registrar.indiana.edu/browser/soc4188/CSCI/CSCI-B649.shtml>
- Topic: Engineering Cloud Computing
- von Laszewski, Gregor
- 3 CR
- RSTR ARR ARR ARR
- Class is taught online
- Discussion (DIS), participation in discussion is mandatory and graded
 - 39143 RSTR 11:15A-12:15P F I2 150

12.0.1.7 ENGR-E 534 Residential

- <http://registrar.indiana.edu/browser/soc4188/ENGR/ENGR-E534.shtml>
- Topic: Big Data Applications
- 3 CR
- RSTR ARR ARR ARR Fox G
- class open to graduates only; class taught online
- Discussion (DIS)
- 14891 RSTR 09:30A-10:45A F I2 130 Fox G
- class meets with INFO-I 423 and I 523

I523 and I423 are the same class

12.0.1.8 INFO-I 523 Residential

- <http://registrar.indiana.edu/browser/soc4188/INFO/INFO-I523.shtml>
- Topic: Big Data Appls & Analytics
- 3 CR
- ARR ARR ARR Fox G
- Class open to graduates only, class taught online

- Discussion (DIS)
- 11857 RSTR 09:30A-10:45A F I2 130 Fox G
- class meets with INFO-I 423 and ENGR-E 534

12.0.1.9 INFO-I 523 Online

- Topic: Big Data Appls & Analytics
- 3 CR
- 11858 RSTR ARR ARR ARR Fox G
- Class open to Data Science majors only
- This is a 100% online class taught by IU Bloomington. No on-campus Class meetings are required. A distance education fee may apply; check your campus bursar website for more information. A distance education fee may apply; check your campus bursar website for more information

12.0.1.10 INFO-I 423 Residential

- Topic: Big Data Appls & Analytics
- <http://registrar.indiana.edu/browser/soc4188/INFO/INFO-I423.shtml>
- 3 CR
- CLSD RSTR ARR ARR ARR Fox G
- class open to undergraduates only, class taught online
- Discussion (DIS)
- CLSD 12403 RSTR 09:30A-10:45A F I2 130 Fox G
- class meets with INFO-I 523 and ENGR-E 534

12.0.2 Syllabus




This document is under construction and will change.

It is obvious that some lectures overlap between different classes as they introduce you or refresh your knowledge about common topics and themes. This especially is valid for aspects dealing with

- Writing Scientific Papers

- Writing Tutorials
- Learning or refreshing your knowledge on python

However it includes also class specific lectures. Both are listed in graphical form in the next two images. The table of content for this and our other documents will provide you easily with the ability to locate them easily and follow the appropriate lectures. It is straight forward. Only thing to remember is that you actually have to read **all** posts on piazza as well as the documents that relate to your class. Links to the documents will be added throughout the semester once they become available.

 *Please note that chapters may be added or removed throughout the semester.*

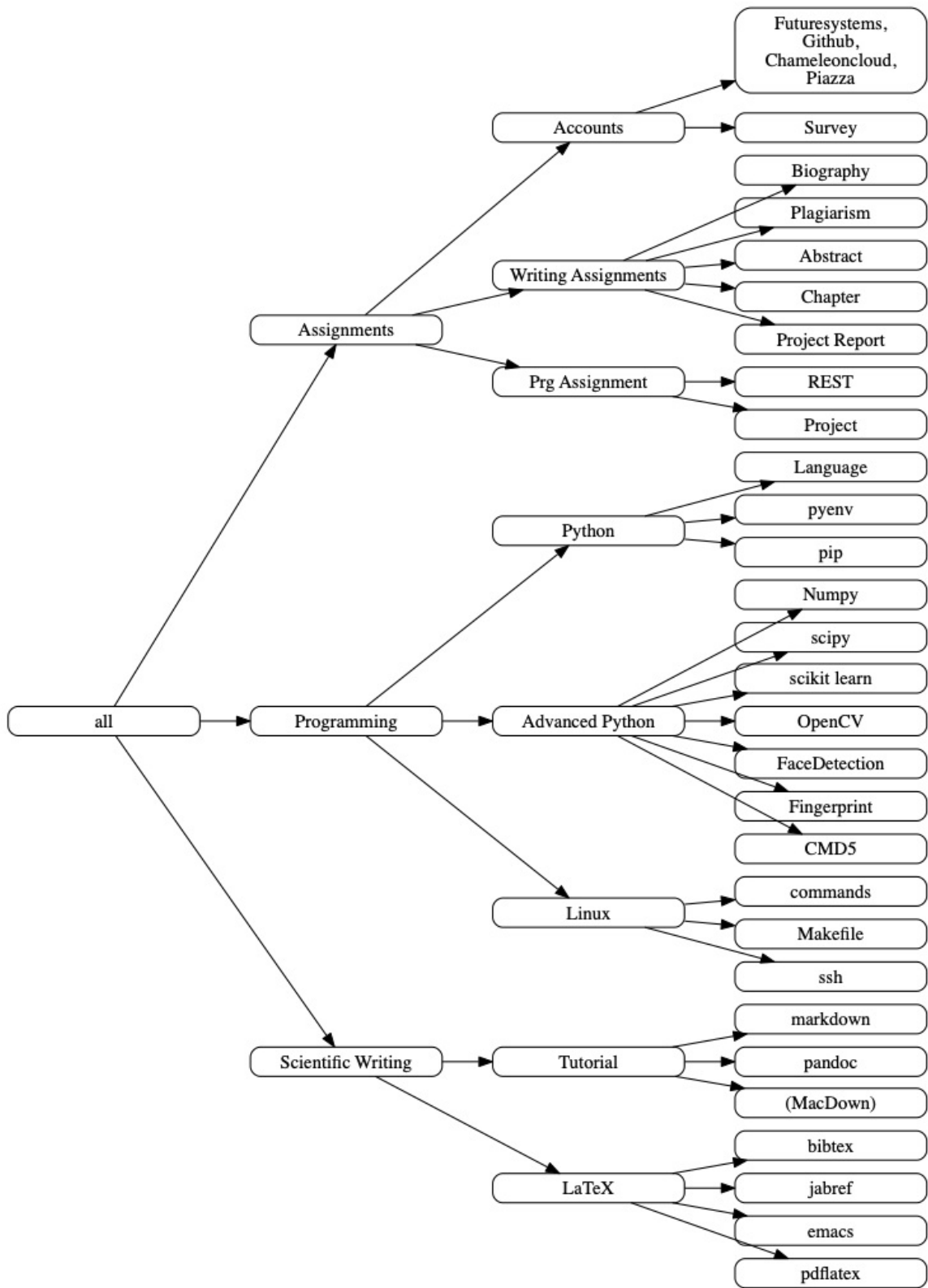


Figure: Common class components

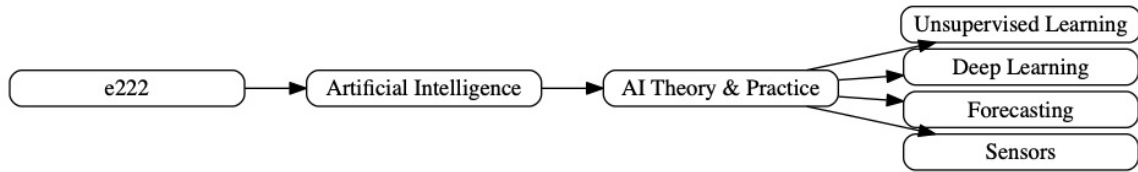


Figure: Class e222 specific components

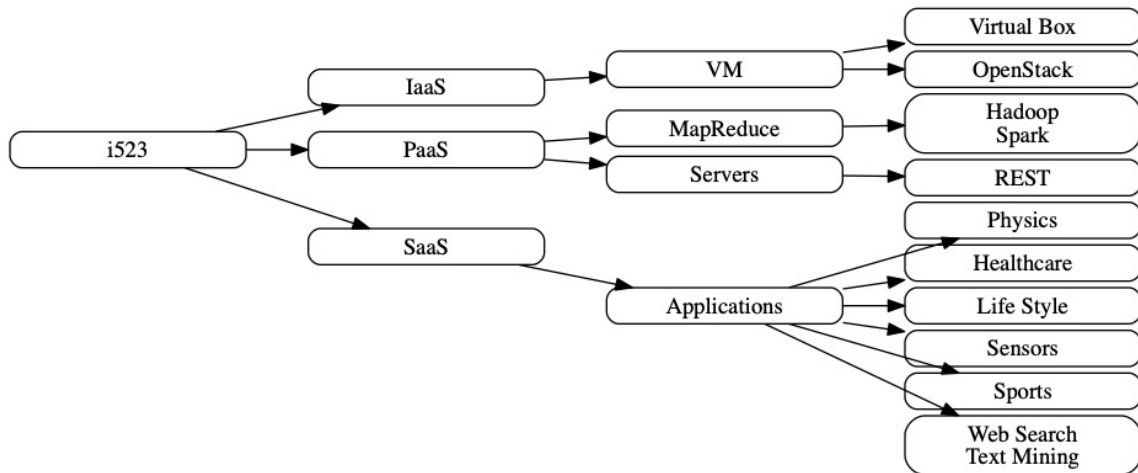


Figure: Class i523 specific components

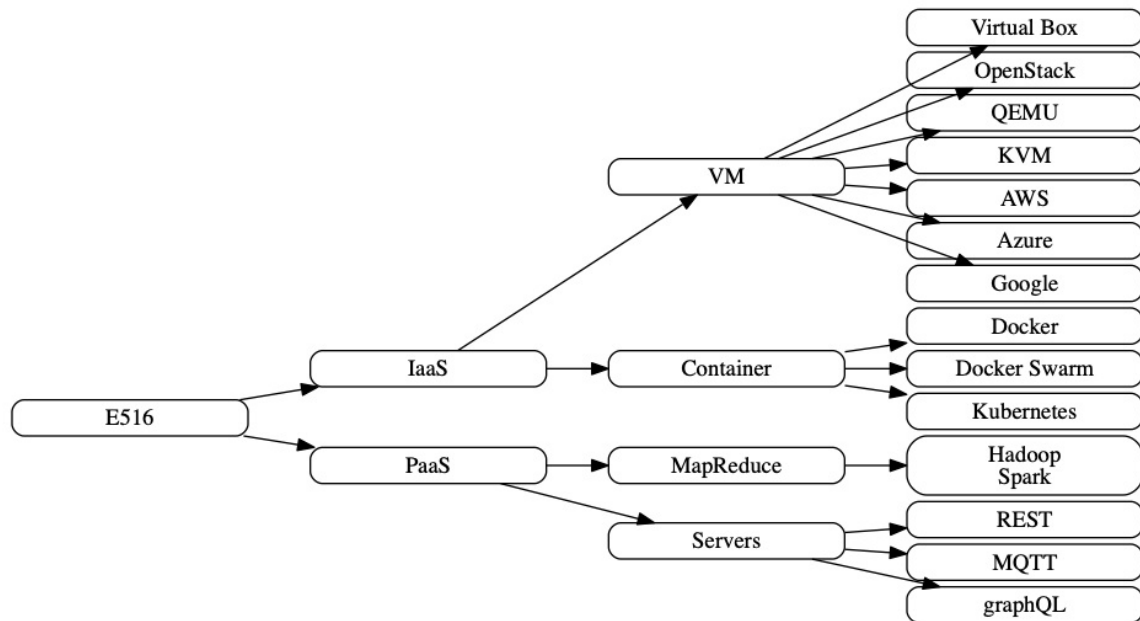


Figure: Class e516 and 616 specific components

12.0.3 Assignments

12.0.3.1 Terminology

Dependent on the class you need to do different assignments. The assignments will be clearly posted. In case of questions, we will update this document to provide clarifications if needed. We use the following terminology:

License:

All projects are developed under an open source license such as Apache 2.0 License. You will be required to add a LICENCE.txt file and if you use other software identify how it can be reused in your project. If your project uses different licenses, please add in a README.md file which packages are used and which license these packages have.

Sections:

Sections are written in markdown and include information on a particular

technical issue that is in general helpful for other students. Sections must be about a substantial topic and include an introduction a section that teaches a reader a significant issue, as well as practical code examples. Multiple small sections can lead to a substantial contribution. We expect that the sections are of high quality and can be included in our handbooks. Please be careful of plagiarism and do not just copy the sections from tutorials or code or from elsewhere.

Technology or Review Paper :

A technology paper is a summary paper about a technology, application, or topic that is not yet covered in other technology papers delivered by previous students of this class. A review paper is a paper that reviews a specific topic related to this class.

In either case includes useful information that provides an overview of what you are trying to describe and analyses its relationship to the class topic. Be mindful about plagiarism. The paper is written in LaTeX or Markdown and uses bibtex for bibliography management. It uses the same format as your report paper. The format is discussed in the Section [Report Format](#).

Note: that for the 2018 we decided to just us Markdown and not LaTeX. We will calculate the exact number of words needed.

Project:

We refer with the term project to the major activity that you chose as part of your class. The default case is an implementation project that requires a *project report* and project code. In case you have issues with code development you can also chose a *term paper* as project.

Term Paper:

A term paper is an enhanced topic paper (only available for I523). The difference is in length and depth of coverage. Comparative or review papers can also become term papers. In case you chose the term paper, you or your team will pick a topic relevant for the class. Term papers should have the quality to be publishable either in a workshop or as part of the handbook.

Not all classes allow you to do a term paper, but require you to do a project. Please confirm with your class. For the classes listed here the term paper will result in a quarter reduction in grade for the entire class not just the paper. Remember tables and figures do not count towards the paper length. A term paper has the following length.

- 8 pages, one student in the project
- 10 pages, two student in the project
- 12 pages, three student in the project

We estimate that a single page is between 1000-1200 words. Please note that for 2018 the format will be markdown, so the word count will be used instead.

Project Report:

A project report is an enhanced topic paper that includes not just the analysis of a topic, but an actual code, with **benchmark** and demonstrated application use. Obviously it is longer than a term paper and includes descriptions about reproducibility of the application. A README is provided that describes in section how others can reproduce your project and run it. Term papers should have the quality to be publishable either in a workshop or as part of the handbook. The format is discussed in the [Section Report Format](#). Remember tables and figures do not count towards the paper length. The following length is required:

- 6 pages, one student in the project
- 8 pages, two students in the project
- 10 pages, three students in the project

We estimate that a single page is between 1000-1200 words. Please note that for 2018 the format will be markdown, so the word count will be used instead.

Project Code:

This is the **documented** and **reproducible** code and scripts that allows a TA do replicate the project. In case you use images they must be created from scratch locally and may not be uploaded to services such as dockerhub. You can however reuse vendor uploaded images such as from

ubuntu or centos. All code, scripts, and documentation must be uploaded to github.com under the class specific github directory.

Data:

Data is to be hosted on IUs google drive if needed. If you have larger data, it should be downloaded from the internet. It is in your responsibility to develop a download program. The data **must** not be stored in github. You will be expected to write a python program that downloads the data.

Work Breakdown:

This is an appendix to the document that describes in detail who did what in the project. This section comes in a new page after the references. It does not count towards the page length of the document. It also includes explicit URLs to the git history that documents the statistics to demonstrate not only one student has worked on the project. If you can not provide such a statistic or all check-ins have been made by a single student, the project has shown that they have not properly used git. Thus points will be deducted from the project. Furthermore, if we detect that a student has not contributed to a project we may invite the student to give a detailed presentation of the project.

Bibliography:

All bibliography has to be provided in a jabref/bibtex file. This is regardless if you use LaTeX or Word. There is **NO EXCEPTION** to this rule. Please be advised doing references right takes some time so you want to do this early. Please note that exports of Endnote or other bibliography management tools do not lead to properly formatted bibtex files, despite they claiming to do so. You will have to clean them up and we recommend to do it the other way around. Manage your bibliography with jabref, and if you like to use it import them to endnote or other tools. Naturally you may have to do some cleanup to. If you use LaTeX and jabref, you have naturally much less work to do. What you chose is up to you.

12.0.3.2 Due dates

For due dates see the [calendar](#) section.

12.0.3.3 Assignment 0 - Organization and Communication

In order for us to communicate with you and you being able to use the class resources we need you to register with various systems. This needs to be done immediately as it takes a while to get the systems set up. It also requires approval steps that take some time and are not immediate, thus do not just start this assignment on the due date. Procrastination will not help you.

12.0.3.3.1 Account setup and survey

Please do the following:

Get access to piazza:

1. We have provided a detailed section about [piazza](#) for you to explain how to get access to piazza. Please review this section carefully and create an account or use your existing piazza account to access the class piazza.

Next obtain some cloud accounts which you may use as part of the class:

2. Obtain an account on <https://github.com/>. Note this is not github.iu.edu and you may have a different account name on the systems. Please write down your *username* on github.com. Your username is not an e-mail.
3. Obtain an account on <https://portal.futuresystems.org/>. Please, write down your username.
4. Obtain an account on <https://www.chameleoncloud.org/>
5. Obtain an account on <https://portal.xsede.org/>. Please write down your username.

Make sure you chose strong passwords that you remember. Once you have access to Piazza do the following:

6. Post a **formal** bio on piazza in the bio folder so you introduce yourself to the class. It also serves as a verification that you have access to piazza and we can communicate with you. An example is available at:

- <https://laszewski.github.io/bio.html>

7. Next fill out the class survey. Only do this after you have created all the accounts and you remember your usernames. The survey is located at

- <https://goo.gl/forms/8Tkf2usxONZIEA182>

After you have submitted the survey we will create your github repository that you will use for your deliverables. This will take some time ...

7. **Wait till your HID directory on github.com is created.** Please do not send us e-mail, we will look at the survey within 1-2 business days (Mon - Fri 9am-5pm EST) we will create your repository on github. Make sure to check on github as you will need to accept the invitation to gain access to the repository. github.com will also send you an e-mail. Check your spam folder. make sure to **meet the deadline** for filling out the survey. It is typically the first Friday of the first week of class. Please be aware of this and conduct this task before Friday.

After your HID directory is created you can continue, with the next tasks. But do not forget to upload them once you have access to the other accounts.

8. Create a notebook.md file in your hid directory on github.com. An example is provided at

- <https://github.com/cloudmesh-community/hid-sample/blob/master/notebook.md>

9. Create a README.yml file in the hid directory on github.com. Make sure to not use any TABS and use valid yaml. If you do not know what yaml is please find out, it is part of this assignment. An example README.yml can be found at:

- <https://github.com/cloudmesh-community/hid-sample/blob/master/README.yml>

You should remove the comments in your README.yml

Naturally, we want you to learn about the various systems. The one you will


need to use immediately is piazza.


10. Learn about Piazza: please see [piazza](#) for more details.

12.0.3.3.2 Class Computer Setup


Now it is time to buy some supplies and prepare your system. Obviously this class will need a computer that you can use to login into the remote machines and do your project. If you do not have such a computer, you can purchase a Raspberry Pi with power plug for about \$50. Otherwise use your laptop or desktop. Detailed information about such hardware are provided in the Raspberry PI section.

Residential students have access to about 100 Raspberry PIs in the Raspberry PI Lab in Smith Research Center (which may be relocated to MESH) Residential students will have mandatory projects for selected classes that will have to be done on the Raspberry PIs. For online students this is optional. Please refer to your specific class.

 *Please note that we do not provide support for Windows. You can certainly use virtualbox and install a newer version of ubuntu on it. Another option is to create a bootable USB stick or external HD on which you install Ubuntu.*

 *Windows 10 Home Edition has significant limitations at time of writing of this document.*

For example it does not allow you to use containers or proper virtualization. For this reason we recommend that you upgrade to Windows 10 pro or education If you like to use Windows. The education version is available for free via IU. Please see (IUware)[<https://iuware.iu.edu/Windows/title/2977>] and share your experience in piazza about your update. You must use the 64 bit version. If your OS doe snot support 64 bit, you need to make sure you update to 64 bit. Please consult the Windows documentation on this. Whatever you do, make a backup first. It may be easier to boot from an external HDD directly into ubuntu.

 *macOS users have typically an easy time as all TAs have access to macOS machines. However the version must be the latest version. We do not support older versions.*

12.0.3.3.3 Outcomes

Obviously this assignment has some implicit tasks and learning outcomes such as

- Learn how to use piazza.
- Learn how to use github.
- Learn how to write a formal bio.
- Set up a computer to be used for class.
- Learn on how to asks for help.
- Work with the fellow students to solve issues and give each other tips.

We expect that if you have difficulties with some of the technologies to also consult with external resources that are for example searchable with google. Using github is mandatory for your project and is a class learning objective.

12.0.3.4 Assignment 1 - Technology

Cloud and big data is an emerging field with lots of activities. To gain an overview of the technologies used in the field you will be tasked to do the following:

1. You will be downloading and studying the technology abstracts that we collected in
 - <https://github.com/cloudmesh/technologies/raw/master/vonLaszewski-cloud-technologies.epub>
2. You will be writing new or improving existing technology abstracts. A technology abstract is not plagiarized, does at most have 30% quotes in it and is between 150 to 450 words long. The text must address what the technology is about and how it is used in cloud and or big data. Each

student will contribute till midterm 10 such technologies. It is important that we make sure that each student has a unique list of technologies they work on. To coordinate this you will be first inspecting the git issues at

- <https://github.com/cloudmesh/technologies/issues>

For each technology that you will add or improve you will add a new issue. You will be using the title

Tech: Technology name

For new entries you will be using the label `new`, for issues that need improvement you will be using the label `open`. You will make sure you assign the issue to you. It is in your responsibility to make sure you are the only and first one that this technology is assigned to. The organization of this is part of your learning experience. You will need to assign this technology to you.

If you find in the git issues als technologies marked with the label `open` they are free for the taking and you can modify the issues and assign them to you. Please also be reminded that all technologies in the technology book that are marked with a red circle need improvements. This is a long list

All this is done on first come first serve.

If you work in groups make sure to assign yourself 10 technologies times the number of team members in your group. Make sure all group members are in github listed as assignee. The maximal number of group members is 3. In fact for this assignment we recommend that you work in groups as this usually increases the quality of of the submission. If you do not have a group find someone to review your submission.

The technologies can be easily modified with pull requests that are managed by the TAs. We suggest that you work with your team and/or reviewer before you create the pull request. Each technology has a small cloud that when you click on it brings you to the github editor so you can create a pull request. Make sure that when working in a group you coordinate the contribution and do not create conflicts. Conflict resolution will be delegated to the team working on the technology.

The ePub publication will be updated typically once a day. So please check if your change is as expected.

The entire technology handbook can be found at

- <https://github.com/cloudmesh/technologies>

12.0.3.4.1 Outcomes

This assignment has the following outcomes

1. Learn about the many technologies as you should not just pick 10 but read up on the one that you are interested in
2. Learn on how to write a small meaning full summary about a technology. This may also be required for you as part of you project. Hence, it is good to know how to do that.
3. Learn using gitissues
4. Learn how to self coordinate lots of tasks in a large group
5. Learn how to create pull requests via the git user interface
6. Learn how to use markdown which will be also used in your project to document how to replicate your project

12.0.3.5 Assignment 2 - Programming Assignment & Chapter

In the handbook we have a large number of practical information that describe a particular topic and include some information to explore this topic with programs and scripts. Your task will be to improve or create a substantial topic of your choice. The topics are included in the following ebooks:

- <https://github.com/cloudmesh/book/raw/master/cloud-clusters/vonlaszewski-cloud-cluster.epub>

The source is located at

- <https://github.com/cloudmesh/book/tree/master/cloud-clusters>

You will be identifying an existing chapter or a new one that you will improve. Just as in Assignment 0 we will use github issues to coordinate this. Note that

this is a different github issue so you need to make sure you use the correct one for the correct task. For this assignment the issues are managed at

- <https://github.com/cloudmesh/book/issues>

For chapter that you work on please use the title

Chapter: Chapter name

Make sure to assign it to you. Use the label `community` to make sure to communicate that this contribution will be done by a community member. As soon as you start working on this and there is text already in the repository, please augment your issue with the url to the document.

TAs will also provide issues that are labeled with `open` that you can take. Once you take it change the label to `community`. Note that the Assignment 2 must be a significant contribution. Multiple smaller ones could add up to a significant contribution.

12.0.3.5.1 Outcomes

This assignment has the following outcomes

1. Inspect the chapters of the handbook
2. Learn how to document code and code execution
3. Learn python, bash, Makefiles and similar
4. Learn how to communicate on how to replicate a code
5. Learn how to benchmark (some contributions may require this)
6. Learn about libraries such as `cmd5` and `click`
7. Learn about Python 2 vs 3
8. Learn about the limitations of anaconda and why we should not use it (valid for some assignments).

12.0.3.6 Assignment 3 - Project

The project deliverables depend on the class you take. Please be aware that the project or term paper constitute to a significant portion of your grade of your class grade. Please locate your class number and read up on the section relevant for your class.

Please see our [Terminology](#) Section about basic requirements. The Report format is discussed in the Scientific Writing Section.

Projects do require you to produce code for all classes the following applies:

Report Format:

All reports will be using the our common format. This format is not the same as the ACM format, so if you use systems such as overleaf or sharelatex, you need to upload it and use it there.

The format for LaTeX and Word found here:

- <https://github.com/cloudmesh-community/hid-sample/tree/master/paper>

There will be **NO EXCEPTION** to this format. In case you are in a team, you can use either github while collaboratively developing the LaTeX document or use MicrosoftOne Drive which allows collaborative editing features. All bibliographical entries must be put into a bibliography manager such as jabref, or Mendeley and exported to bibtex. This will guarantee that you follow proper citation styles. You can use either ACM or IEEE reference styles. Your final submission will include the bibliography file as a separate document.



Documents that do not follow the ACM format and are not accompanied by references managed with jabref or are not spell checked will be returned without review.

Code:

You must deliver the **source code** in github. The code must be compilable and a TA may try to replicate to run your code. You **MUST** avoid lengthy install descriptions and everything must be installable from the command line. We will check submission. All team members must be responsible for one or all parts of the project.

Code repositories are for code, if you have additional libraries that are

needed you need to develop a script or use a DevOps framework to install such software. Thus zip files and `.class`, `.o` files are not permissible in the project. Each project must be reproducible with a simple script. An example is:

```
git clone ....
make install
make run
make view
```

Which would use a simple make file to install, run, and view the results. You are expected to integrate `cmd5`, which we teach in class. In addition you can use or are expected to use `DOCKERFILES`, `ansible`, or shell scripts. It is not permissible to use GUI based DevOps pre-installed frameworks. Everything must be installable and reproducible from the command line.

Cloud Resources:

Projects may be executed on your local computer, a cloud or other resources you may have access to. This may include:

- chameleoncloud.org
- furturesystems.org dockerswarm
- furturesystems.org kubernetes
- AWS (you will be responsible for charges)
- Azure (you will be responsible for charges)
- virtualbox if you have a powerful computer and like to prototype
- Raspberry PI's
- other clouds, please confirm with us.

Access to clouds must be scripted and a `cmd5` extension must be developed as part of your project to receive full credit. You must not just use your local PC as you need to use at least one cloud. If you work in a team each team member needs to work at least on one cloud.

12.0.3.6.1 Class: E516 or E616

The objective of the project is to define a clear problem statement and create a framework to address that problem. This framework must include a docker packaged service that includes all necessary dependencies necessary to perform

the analysis. For example if you are using a machine learning algorithm your docker packaged service will include the machine learning algorithm which works on a particular dataset providing clustered ,classified or regression outputs to the user. The final objective is to obtain results in form of graphs or tabular mode. In selecting a dataset, you can use a dataset from a public dataset like and pick a suitable dataset according to your problem statement. Final results must be exposed via a REST service. For instance if it is a classification problem, the should have the capability of entering a new test data set or single data point (meaning a single record) via a REST API endpoint and get the expected output in form of a JSON object. UI creation is an optional task. This is the overall expectation of the project.

12.0.3.6.1.1 Deliverables

- Find a data set with reasonable size (this may depend on your resources and needs to include a benchmark in your paper for justification).
- Clean up the data set or make it smaller or find a bigger data set
- Identify existing algorithms and tools and technologies that you can use to analyze your data
- Develop a Swagger or Flask Rest Service to send a sample data set and get output. Provide benchmarks.
- Take results in two different cloud services and your local PC (ex: Chameleon Cloud, echo kubernetes). Make sure your system can be created and deployed based on your documentation.
- Create a Makefile with the tags deploy, run, kill, view, clean that deploys your environment, runs application, kills it, views the result and cleans up after wards. You are allowed to have different makefiles for the different clouds and different directories. Keep the code and directory structure clean and document how to reproduce your results.
- For python use a requirements.txt file also
- For docker use a Dockerfile also

- Write a report that includes the following sections
 - Abstract
 - Introduction
 - Architecture
 - Implementation
 - Technologies Used
 - Results
 - Deployment Benchmarks
 - Application Benchmarks
 - (Limitations)
 - Conclusion
 - (Work Breakdown)

12.0.3.6.2 Class: i423 i523 or E534 or i524

The objective of the project is to define a clear problem statement and create a framework to address that problem as it relates to big data your project must address the reproducibility of the deployment and the application. A dataset must be chosen and you can analyze the data. While in i423, i523, the data analysis has more weight, in e523 and i524 you must also address the deployment. For the later classes see also the project requirements of E616.

You have plenty of time to make this choice and if you find you struggle with programming you may want to consider a term paper instead of a project.

In case you chose a project your maximum grade for the entire class could be an A+. However, an A+ project must be truly outstanding and include an exceptional project report. Such a project and report will have the potential quality of being able to be published in a conference.

In case you chose a term Paper for I524 your maximum grade for the *entire* class will be an A-.

12.0.3.6.2.1 Deliverables

- Find a data set with reasonable size (this may depend on your resources and needs to include a benchmark in your paper for justification).

- Clean up the data set or make it smaller or find a bigger data set
- Identify existing algorithms and tools and technologies that you can use to analyze your data
- Develop a Swagger or Flask Rest Service to send a sample data set and get output. Provide benchmarks.
- Take results in two different cloud services and your local PC (ex: Chameleon Cloud, echo kubernetes). Make sure your system can be created and deployed based on your documentation.
- Create a Makefile with the tags deploy, run, kill, view, clean that deploys your environment, runs application, kills it, views the result and cleans up afterwards. You are allowed to have different makefiles for the different clouds and different directories. Keep the code and directory structure clean and document how to reproduce your results.
- For python use a requirements.txt file also
- For docker use a Dockerfile also
- Write a report that includes the following sections
 - Abstract
 - Introduction
 - Architecture
 - Implementation
 - Technologies Used
 - Results
 - Deployment Benchmarks
 - Application Benchmarks
 - (Limitations)
 - Conclusion
 - (Work Breakdown)

For the final project in this class you need to do the following.

- Find a dataset : A sample data repository can be found at the link next however you are free to use data from elsewhere, like scikit learn libraries.
- Use Scikit Learn Library to run Machine Learning algorithm depending on your problem.
- Do some classifications, clustering or a regression calculation using a machine learning algorithm.
- Use the results from classification to draw charts. You can use matplotlib to do this.
- Use REST api to tun ML algorithm i.e. in k-means the user should be able to change the cluster number (k) through a RESTful service.
- Use Flask Rest API to expose the data to the viewers. So people can send a data set and get the outputs as a json object.

12.0.3.6.3.1 Deliverables

- Find and clean up data set
- Scikit Learn ML Algorithm Application to cleaned up data set
- Clustering or Regression or Classification results in Graphical Format (Matplotlib) (GUI not needed, save graphs as png)
- Flask Rest service to tune ML algorithm
- Flask Rest Service to send a sample data set and get output (terminal output is enough)
- Create a Makefile with running, setting up, sample test case running commands
- requirement.txt file for pip installation

- Package with Docker

- Write a report that includes the following sections
 - Abstract
 - Introduction
 - Architecture
 - Implementation
 - Technologies Used
 - Results
 - Deployment Benchmarks
 - Application Benchmarks
 - (Limitations)
 - Conclusion
 - (Work Breakdown)

12.1 GRADING

Grading for homework will be done within reasonable time of the submission if the submission was on time. This however could still take multiple weeks. Students that miss the deadline will be graded on best effort, which could mean at the end of the semester as TAs will grade first papers that have been handed in on time. A fractional grade reduction will be given for residential students if the project is late (e.g. A becomes A- and so on). Some homework can not be delivered late (which will be clearly marked and 0 points will be given if late; these are mostly related to setting up your account and communicating to us your account names such as Assignment 0.)

It is the student's responsibility to upload submissions well ahead of the deadline to avoid last minute problems with network connectivity, browser crashes, cloud issues, etc. It is a very good idea to make early submissions and then upload updates as the deadline approaches; we will grade the last submission received before the deadline. To allow TAs to monitor your progress we require a weekly checkin.

Note that the assignments in this class will take a considerable amount of time and doing proper time management is a must for this class. Avoid starting your

project late. Procrastination does not pay off. Starting a paper a day or even in the week before the deadline will allow you not to achieve your best. Late Projects or term papers will receive a 10% grade reduction.

If we see it necessary additional assignments will be added. Instead of tests the assignments have build in checks for you understanding the topics.

12.1.1 Grades on Canvas

The final grade for your class is NOT accurately posted in CANVAS. Please visit the registrar for your final grade.

We have run many times into issues with CANVAS thus we try to stay as much as possible away form it. We know that the total grade will not be accurately reported in CANVAS. Thus please do not contact us check your grade in the registrar.

CANVAS has also the tendency to give the students a wrong sense of security as all non graded assignments are not included into the grade calculation on CANVAS. This again, your grade will be wrongly reported by CANVAS.

Furthermore we are using only a letter grading scheme that distinguishes qualitatively between grades. Typically we do not engage in arguing if you get a point more or less. Instead we look at the artifact and decide if it is an A or A- and so on. CANVAS on the other hand uses a point system that may provide a misleading information as we do not use points.

Once the due date is passed all incomplete assignments will appear as an *F* in CANVAS. Once we receive and review the submission this grade will be changed. Please, do not contact us if you submitted late and you see an *F* in CANVAS temporarily.

In case you get an incomplete for that class, all grades in CANVAS stay as *F*. You will need to look at the registrar grade

12.1.2 Discussion about Grades

Should it be necessary a discussion about a grade must not be taking place via e-

mail nor via piazza. You must use a private post but post it to the instructors only.

12.1.3 Project Ideas

A good way to start identifying a project is to look at previous student projects. For this reason we provide here a list of class proceedings that have been authored by previous students of cloud classes. Please remember that the proceedings could include papers that do not fulfill the class requirements or are incomplete. It is up to you to identify excellent project and use them as a yardstick.

Volumes include (with highlighted contributions, there are many more so please look at the volumes):

2. [Use Cases in Big Data Software and Analytics Vol. 3, Gregor von Laszewski, Fall 2017](#) (426 pages)
 - Predicting Housing Prices, Murali Cheruvu, Anand Sriramulu
 - There are many other good examples in this volume. This volume focusses more on Analysing and identifying a big data problem. Also it includes some nice Raspberry PI applications.
3. [Projects in Big Data Software and Applications, Gregor von Laszewski, Spring 2017](#) (196 pages)
 - *Automated Sharded MongoDB Deployment and Benchmarking for Big Data Analysis*, Mark McCombe, Gregor von Laszewski
4. [Cloud and Big Data Technologies Vol 9, Gregor von Laszewski, Spring 2018](#) (189 pages)
 - *Location based crime data search and analysis with Spark UDF*, Kadupitiya Kadupitige, Gregor von Laszewski
 - *Big Data Reference Architecture using Python Celery*, Sabra Ossen, Gregor von Laszewski
 - *Benchmarking a Sentiment Analysis Algorithm Using Hadoop on Multiple Platforms*, Min Chen, Bertolt Sobolik, Gregor von Laszewski
 - *Leveraging REST for cloud portability*, Michael Robinson, Sushant Athaley, Harshad Pitkar

One important policy we have in the class is that all projects must be unique from each other. Thus you should identify different data sets, algorithms or

infrastructure from all current and previously produced reports. That should not be an issue as there is a lot of different opportunities for you. Finding a unique project is part of the class goal.

There will be a number of project topics available that are of interest to us. However, it is your responsibility to expand them to make them project worthy.

The paper format is included in hid-sample project-report (e.g. same as paper):

- <https://github.com/cloudmesh-community/hid-sample/tree/master/project-paper> The code and the paper are to be added in your hid folder. You will be creating lower case directories called `project-paper` and `project-code`. You will not check in any data, but instead create scripts that fetch the data.

For an example directory structure, please see

- <https://github.com/cloudmesh-community/hid-sample> Certainly, you can choose from many different topics and we hope you pick one that is suitable for you and you enjoy doing. You have the opportunity to definitely pick a project that you enjoy doing. However it must be related to the course. This course is not about finding the best algorithm or copying a project from your AI or other cloud classes you have taken at IU. It is about finding a novel project that is related to cloud computing, big data and the deployment of the system on cloud resources.

12.1.3.1 Project Data Restrictions

We will not accept any project using the Titanic, Wordcount. Furthermore we observed that many students that used Kaggle datasets did not focus on deployment issues, but only on data set analysis. Please remember you need to deploy your code on a cloud and make that deployment reproducible. It is not sufficient to show that you are the only one being able to run your programs.

12.1.3.2 Register Your Idea

Please register your project idea you need to do two things. First you file a github issue with the title

You assign it to all team members in your project. An abstract is submitted in that issue that describes what you do in your project. The course number is the number of your course. We will be commenting on your project idea. It is your responsibility to see if this project has already been done. Please do not just go ahead and copy someone else's code, try to do something that you do. However you can leverage toolkits such as scikit learn, tensorflow, and others.

12.1.3.3 Example Ideas

Next we list some project ideas that are suitable for the classes. At this time we are especially interested in creating a big data reference architecture. Thus your project could be incredibly valuable to contribute to this. This is done in collaboration with NIST. Other projects include the deployment of cloud clusters. This might include the following technology tech

1. Slurm/mpi
2. Docker
3. Docker swarm
4. Kubernetes
5. Ssh worker
6. OpenFaas
7. OpenWhisk
8. Hadoop
9. Spark.

Naturally they need to be *reproducible* deployments. To test out such projects, you will also need to demonstrate a benchmark on an application.

We provide some more details next.

12.1.3.3.1 Project Type A: NIST Rest Services Project

This project idea is the simplest one of the once listed in this section as we have extensively discussed it and provided all important information to succeed in this activity. However, this task must not be underestimated as it requires some non-trivial work as any of our other tasks. We believe however that the efforts for it

is smaller than with other project ideas.

We have provides you with the NIST big data reference architecture. As part of this we have identified how to create rest services. In this project you will define a **SIGNIFICANT** set of resources and implement the rest services for them. IN contrast to your previous assignment this is a set of services and not just a single service. (for example just implementing a key value store abstraction is not sufficient). A proper project scope includes for example an abstraction of resources related to VMs on AWS with libcloud, or VMs on Openstack with libcloud, or VMs on Azure with libcloud, or An abstraction for data storage (not just files, but also objects and key value pairs), the abstraction of an accounting framework, and so forth.

In case you have not completed your swagger REST service a portion of this project will be used to satisfy that requirement.

One nice project would for example be the automated creation of rest services while using a function specification of python. This way you could for example look at scikit learn, write 10 use cases, use your code generator and create for each of them the rest service. Important would be a scalability test.

12.1.3.3.2 Project Type B: Reproducible Raspberry PI Projects

The raspberry PI projects are divided topically by class. In this project you will be developing or leveraging form an existing tutorial developed as part of the class. You will be focusing on how to create for each tem member on one of the following technologies

1. Slurm/mpi
2. Docker
3. Docker swarm
4. Kubernetes
5. Ssh worker
6. OpenFaas
7. OpenWhisk
8. Hadoop
9. Spark

You will create a reproducible deployment and work towards the implementation of a deployment. In previous classes students focused on setting this up for a small number of nodes. What we need to do now is to expand this to a scalable solution with many hundreds of Pi's in the cluster. Naturally login in by hand on these machines is not suitable, but you need to automatize this process as much as possible. Your ideas on how to do this are most welcome. There are different strategies, such as burning all SD cards with a program on your laptop and modifying the file system of the sd card after the burning, setting up a simple minimal system with ssh enabled and DHCP so you can log into a named host and use parallel commands to further provision the system, or even PXE boot. Once you have figured out and documented this you will be documented how to deploy a Hadoop and/or a spark cluster on the Pis.

You will then pick a data set and do a application and measure the performance.

In case you work in a team, each person in the team needs to add a new deployment. Example, if you are in a 3 person team you need to do not only do a single deployment but multiple. This could even mean that you need to deploy it on echo which is a non Pi cluster, but you can get great performance comparisons between your analysis on echo and the one on the PI. Other examples could include the comparison of spark with hadoop on PI and echo

As this project contains a significant of tutorial like activities (just do not use the term tutorial, but in this section we describe) we recommend that you develop the setup procedure in markdown. and not directly in latex. However use *clean* markdown and follow the markdown rules. We have seen in the tutorial to be delivered for this class many wrong examples on how to not use markdown.

For this reason the length of the paper may be reduced by one page if the set up procedure is excellent, and includes automated deployment scripts with minimal input by hand (this requires programming).

12.1.3.3.3 Project Type C: Reproducible Data Analysis

This project requires you to use one cloud IaaS resource such as chameleon, Futuresystems Echo, AWS, or Azure. You will be deploying on the IaaS and conducting based on a data set that you conduct an analysis of the data. You will be benchmarking the time it takes to set up this environment as well as

benchmarking how fast the analysis is.

12.1.3.3.4 Project Type D: Define Your Own

Define your own project and discuss with us in the Monday meeting with Gregor. A good example is a student that has chosen GraphQL as the major infrastructure component. He is developing a contributed chapter for the handbook, a tutorial, and a deployment and benchmark of data of his choice.

12.1.3.4 Special Projects

Some projects that you may be interested in and should be done throughout the entire semester in order to be successfully. This includes making sure that Assignment 0 - 3 all integrate with each other.

12.1.3.4.1 Project Cloud Security

Please read up on security and attribute base security. Take a look at the already developed Web services to showcase how we develop flask and swagger servers with basic auth (needed to understand the attribute based security).

- https://en.wikipedia.org/wiki/Attribute-based_access_control

This project has three parts and could be used throughout the class for all assignments.

Technologies

Identify and summarize technologies related to cloud or big data and security. Do more than 10.

Paper

Survey of Attribute based security and other security for clouds pages = number of people * 2 maximal 3 people (no images as usual in page number. Integrate this in a general overview about cloud, big data and security<

Tutorial

find frameworks in Python that do this. If they exist to develop a tutorial

Swagger

develop a swagger rest service managing the attributes and entities in the framework

Alternative A: Project VM based.

showcase this in a project that does this in a cloud framework using distributed virtual machines and services. Develop a tool that autogenerates services based on a function definition while also adding attribute based security.

Alternative B. Container based

do the same project but instead of using VMs do it in containers.

This project has enormous potential as (a) NIST Is highly interested in this. Publication potential of one or two papers. (b) security is hot, and (c) cloud is hot.

12.1.3.4.2 Hadoop 3.0

Hadoop was recently updated to Hadoop 3.0. Develop reproducible deployment scripts for a variety of infrastructures (container, PI, OpenStack, AWS, Azure, google, ...)

Start with docker.

- <https://github.com/sequenceiq/hadoop-docker/blob/master/Dockerfile>

Max 3 people can work on this, while deploying it on 3 platforms and showcasing it works with a benchmark.

Expand this project to do benchmarking on various infrastructures. This project is best done in a team of 2, do as many infrastructures as you can achieve. Keep

the benchmark simple. Develop cloudmesh commands to interface with it. If time allows develop rest services.

12.1.4 Piazza

We use Piazza (<https://piazza.com>) because questions and answers on Piazza are community-edited and provides the opportunity not only for instructors, but also for students to contribute. Each question has a single answer edited by the students of the class and if needed an instructors' answer that is collaboratively edited by the instructors.

Due to this wiki-style question and answer, when a student has a question, one does not have to look through long e-mail threads but instead can look at the answer. For details that lead up to the answer you are highly encouraged to also look at some comments that lead up to the answer.

An advertisement video from Piazza summarizes the features:

-  [Piazza Overview from Piazza](#)

Piazza Support with a lot of information is available at:

- <http://support.piazza.com>

A good document about piazza is available at

- https://piazza.com/pdfs/piazza_product_introduction.pdf

12.1.4.1 Access to Piazza from Canvas

Piazza is one of the recommended IU supported technologies within CANVAS. It replaces the CANVAS discussion groups with superior technology targeted to support large student classes while also focusing on student engagement.

To access piazza you can have the following situations provided in the next four subsections. Please read **ALL** of them **CAREFULLY**, decide which applies to you and follow the instructions. If you have improvements to this instructions, please let us know.

12.1.4.2 Piazza for 516

Due to a bug in canvas the piazza location for 516 could not be crosslisted to CANVAS. Thus we simply add the following link to it

- <https://piazza.com/iu/fall2018/516>

This link is used also for the undergraduates and the CS students

If you have any issues with enrolling in Piazza, please contact us in the office hours.

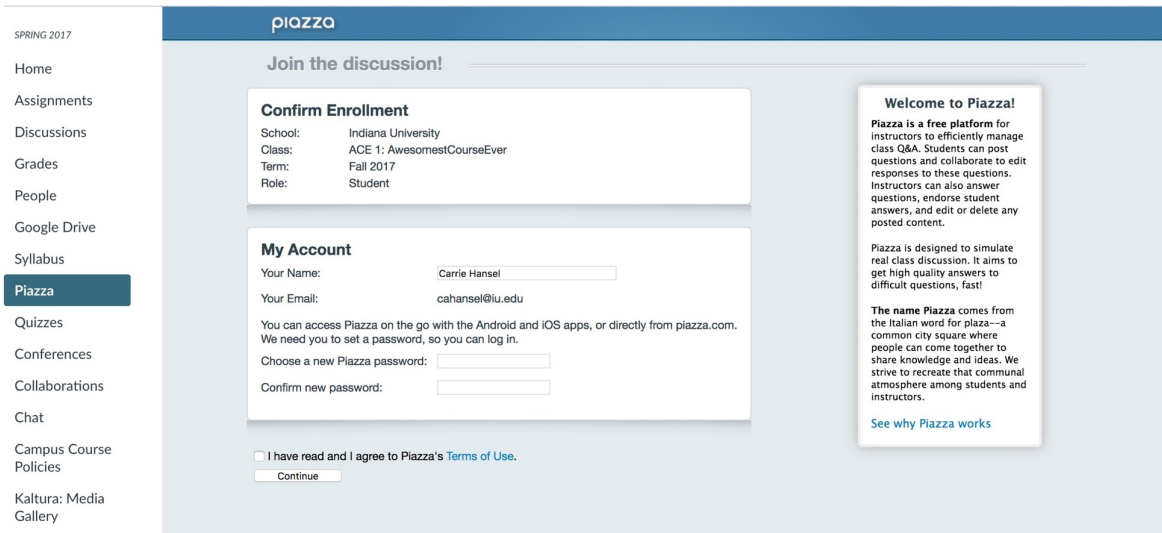
In Piazza we always use the @iu.edu account.

The accounts `yourid@indiana.edu` and `yourid@iu.edu`, even though they are they organizational IU accounts, are considered as two different accounts for Piazza. If you have used in piazza previously the @indiana.edu account, you will need to use the @iu.edu account instead as this is the preferred account for external services that IU offers. To do that please go to piazza and then go to the settings within piazza. Here you add your @iu.edu and merge it with your existing @indiana.edu account. This can also be done with any other account you may have used in case it was not the @iu.edu account.

12.1.4.3 Piazza for other classes

12.1.4.3.1 Situation: You have never logged into piazza

First, Click the Piazza link on the left navigation of your Canvas course.



image

Second, create password and accept terms.

The email address shown on this screen is your default IU email address. It is the address Canvas sends to all integrated tools like Piazza. You can not edit it, so do not try.

The password you create here is for accessing Piazza from a mobile device. You must use the default IU email address from this screen to access this account on another device, so make a note of it.

My Account

Your Name:

Your Email:

You can access Piazza on the go with the Android and iOS apps, or directly from piazza.com. We need you to set a password, so you can log in.

Choose a new Piazza password:

Confirm new password:

I have read and I agree to Piazza's [Terms of Use](#).

image

Choose current degree program (only important if you want to opt into their recruiting program on the next screen; choose whatever you want here)

Third, associate your IU account

image

Forth, if all goes well you see the Success screen

image

12.1.4.3.2 Situation: You have logged into piazza and used your default IU e-mail

1. Click the Piazza link on the left navigation of your Canvas course.
2. You will be automatically enrolled in the course Piazza site and logged in.
3. Start using Piazza.

12.1.4.3.3 Situation: You have logged into piazza and you used another non IU e-mail

1. Click the Piazza link on the left navigation of your Canvas course.
2. Proceed as in #1. This will create your new Piazza account that is linked to your courses in Canvas. This is the account you should always use in your IU courses.
3. If you wish to merge other accounts that you own, please see [Add an email address or merge two accounts](#).

12.1.4.3.4 Situation: You have multiple accounts in piazza

1. If one of your multiple accounts corresponds with your default IU email address, you will be automatically enrolled in the course Piazza site and logged in.
2. If none of your accounts corresponds to your default IU email address, follow the instructions in #3.
3. If you wish to merge other accounts that you own, please see [Add an email address or merge two accounts](#).

I post the official response from the CANVAS team here: “When a student clicks the Piazza link in your course navigation, they will be authenticated through to Piazza. If the student already has a Piazza account that matches their default Canvas email, they will simply be enrolled in the Piazza course. If the student does not have an account, Canvas sends the pertinent information (default email address primarily) to Piazza, Piazza creates the student’s account and enrolls the student in the Piazza course. There is nothing you need to do to.”

If you have any questions regarding accessing piazza, please send them to

“Ricci, Margaret P” <>

12.1.4.4 Verify you are on Piazza via a post

Post in the **bio** folder a short introduction about yourself. One that you could include in a paper.

An example is provided at <https://laszewski.github.io/bio.html> with an image at <https://laszewski.github.io/images/gregor.jpg>

Use the subject line *Biography: Firstname Lastname* and post it into the bio

folder.

12.1.4.5 Making Piazza Work

In order for Piazza to work students and instructors need to participate

Students participate: Students must collaboratively work on an answer to a question. Students must not post irrelevant followups to a question. If you notice your comment was irrelevant, please delete it. Students must **search** prior to asking a new question if the question has already been asked. Duplicated questions can be merged.

Instructors guide: The instructor guides the students in order to obtain an answer to a question. In some cases the instructor may be the only one knowing the answer in which case he tries to provide it.

Not using e-mail: Instructors will and must not use e-mail to communicate with a student. All communication will be done via piazza. There, are only very view situations where e-mail is allowed, ask on piazza first if you should engage in e-mail conversations.

Not using CANVAS discussions: We will not engage in any CANVAS message exchange. Any communication is to be done on Piazza. It is in your responsibility to enroll in Piazza to make it work for you. Instructions are posted in this document. (Grade discussion will be done in CANVAS)

12.1.4.6 Towards good questions

Naturally when you ask a question you need to do it in a reasonable form and provide sufficient information so that the question can be answered. It is in the responsibility of the student to update the question to provide enough information.

Thus information may include: Firstname Lastname, HID, and URL to document in question

To give you an example of a **bad** question consider:

send from Xi Lee

Hi Professor:

I read a nice article about apples
and potato's and updated my
paper. Please give me feedback

Thank you

Kevin

Here the reasons why this can be improved:

1. As professors and instructors may review your document it is unnecessary to start with “Hi Professor:”, just leave it away. If you want a particular instructor use the name explicitly, such as “Gregor:”, e.g. multiple professors may be teaching your course.
2. You have not specified which article you read, you need to include the **URL** to the article so we can follow your argument.
3. You have not included the **URL** to your document so we do not know what you are talking about. Remember there are many others students in the class.
4. You are using a different name from the one that you are registered with. This can lead to confusion when we look up your name. We prefer that you use only one name that is associated with your e-mail.

The previous bad question will simply be commented on (if at all):

“Missing information” or “?” indicating that information is missing.

It is in your responsibility to figure out which information is missing. You need to modify the original post and.

12.1.4.7 Guide on how to ask good questions

This guide is adapted from

- <http://www.techsupportalert.com/content/how-ask-question-when-you-want-technical-help.htm>

Steps to getting your question answered on piazza

1. Before you even go to ask a question, think through what your problem is.


Write down how you are going to describe it. Think about it from the other side - what would you need to know if a student came to you and asked the question? Gather all the system information that seems to bear on the problem (see how at this link). Sometimes it even happens that by thinking through the problem, you come up with the answer yourself.

2. Verify that your question has not yet been answered with a search on the Web, Class Web page, or class piazza, this may require multiple searches.
3. In case it is a technical question, write down any error codes that appear on your screen. Do **not use screenshots** if the text is characters. This is because a reply may need to paste and copy from the original. Also screenshots are not searchable. We will not answer any questions that post screenshots if they are not necessary. It is far easier to copy and paste and use terminal type in the formatting. Also if the text is posted it is searchable. (Any unnecessary screenshot will receive a point deduction. Based on experience we have to do this as previous students in other classes ignored this policy).
4. Place your question or problem in a forum that is relevant to its subject. That may seem obvious but anyone who has experience with forums knows that a lot of questions show up in the wrong place. You will need to identify one or more a fitting piazza “folders” (folders sort the posts by topics).
5. Select a title that briefly and accurately describes your problem. A title like “Help!” or “Computer will not work” will often get ignored. Almost any problem can be titled with a few key words that will raise interest in somebody who is familiar with the subject. A corollary to this is to avoid using all caps or a lot of exclamation points. Something like “HELP!!!” turns many people off.
6. In the post, briefly describe the problem in a paragraph. Leave out unnecessary details. Save everybody time by listing any solutions that you have tried but did not work. Avoid using screenshots if they are not needed. (I mention this again).
7. In case of a technical issue describe relevant system details. For example, it is essential to designate your operating system and type of computer and

any components that might be involved in your problem. List any error code that has been displayed. Be prepared to provide more details if asked.


8. Tell what you were doing when you encountered the problem. If it is a reproducible problem, list the steps or computer operations that cause the problem.
9. If applicable, List any recent software you have installed or hardware changes you have made. If you have updated any drivers recently, also list that.
10. Formulate your questions and answers in a courteous manner. Respect the answers from others. Somebody is giving you their time and expertise for free. You may want to come back to the forum and it pays to be friendly.
11. If a suggested solution works, be sure to return to piazza and report your success. It is the least you can do to return something for the help you have been given. It will make you welcome in the forum the next time you go there for help.

12.1.4.8 Piazza class Links

 *Using the direct links listed in the next two sections, can lead to you not getting proper access via Canvas. If you click on these links **before they create** the account via the link in your current Canvas course, you will create an account that is not matched up with Canvas. To avoid issues make sure you integrate to piazza via Canvas first.*

If you have questions about this contact Margaret Ricci (mricci@iu.edu).

12.1.4.8.1 Current Classes

 *Before clicking on the links you need to have an account created.*

Class	Semester	Year	Link
E516	Fall	2018	piazza home link

12.1.4.8.2 Previous Classes

Class	Semester	Year	Link
E516	Spring	2018	piazza home link
E616	Spring	2018	piazza home link
I524	Spring	2018	piazza home link
I523	Fall	2017	piazza home link
I524	Spring	2017	piazza home link
I523	Fall	2016	piazza home link

12.1.4.9 Piazza Curation

We are using Piazza in a curated fashion. This means that we try to file posts into folders and delete messages that are not relevant or do not provide any additional information. It also means that if you see an error in the post that students and instructors have provided, you should improve it. This may include spelling errors.

As part of the curation we are introducing a number of folders. The folders may vary from class and we can add new folders if needed. Please let us know.

We provide next a list of example folders that we have used in previous classes and list their purpose. Please help us to add folders from your class.

Folder	Description
logistics	Any question and discussion related to the logistics of the course.
lectures	Any question and discussion related to the lectures.
project	Any question and discussion related to the project.
chapter	Any question and discussion related to the chapter assignment
section A	ny question and discussion related to the section assignment.

python	Any question and discussion related to python.
pi	Any question and discussion related to the Raspberry Pi 3. We are not using older Raspberry Pi's and therefore can not comment to them.
bio	A homework folder in which you only publish your bio. The bio needs to be published as a <i>note</i> . This assignment also serves us to see if you are in piazza. Please do this assignment ASAP. You need to post a formal bio. See the many great examples in the folder.
help	If you need help and none of the other folders fits, please use this folder. If information from here will result into new Web page content it will be added and marked into the folder <i>resolved</i> . See the <i>resolved</i> folder for more detail.
resolved	Sometimes we move some general help messages to the resolved folder in case the help message results into information that is posted on our class Web page. We than will add a link to where in the class Web page this question was answered. The TAs will aggressively try to put information into the Web page.
discussion	Any content that deserves its separate discussion and is not covered in the listed folders.

In addition to these general folders we also have some folders which **MUST NOT BE USED BY ANY STUDENT TO POST CONTENT**. These folders serve to communicate your assignments and are used internally between Gregor and the TA's. Please do not mark any questions about assignments with the `assignments` folder, but instead use `help`.

assignments:


This folder only lists the assignments. At any time in the class you can click on the assignment folder and list the assignments given to the class. Thus there is no confusion which assignments have been given. In case students have questions about assignments they should not use the *assignments* folder, but the *help* folder. TAs are instructed to correct wrongly filed messages in folders.

In case you decide to post privately and the information is useful for others also,

the message will be published to the class.

12.1.4.10 Read the Originals, not just the e-mail

Piazza provides a convenient mechanism to update you through e-mail when an answer is changed or when someone posts.

 *In many cases additional information is available for the post and it is **not sufficient to just read the mail.***

The mail is just a reminder that something happened. Use the `<click here>` feature in the mail to get not only to the update, but to the actual post. Then you can get reminded about the information that is part of the post and potentially answers your question in full. It is not sufficient to participate in this class while only reading email, you should participate while visiting piazza and actively contribute to it.

12.1.4.11 Exercises

Piazza.1

Enroll in piazza

Piazza.2

Post a short formal bio in the bio folder and optionally include a professional portrait of yourself. Make sure you understand what a formal bio and portrait is. Research this in the internet. Look at IEEE papers for examples.

Piazza.3

How do you find out within Piazza which assignments have been posted?

Piazza.4

Please watch the Video about Piazza

Piazza.5

Why is it important to not just read the piazza e-mail, but to visit the post by using the [click here](#) feature in the mail?

12.1.5 Class Git

This class will use git to manage all assignment submissions. We use the publicly available github.com. The class git is hosted at

- <https://github.com/cloudmesh-community>

It is in the responsibility of the student to create a github account and make sure that you will be added to the class github within one week of joining the class. Make sure to fill out the survey to communicate the github.com username.

Previous github locations include:

- <https://github.com/cloudmesh-community>
- https://gitlab.com/cloudmesh_fall2016
- <https://github.com/bigdata-i523>

Previous book githubs include

- <https://github.com/cloudmesh/book>

13 APPENDIX

13.1 FAQ

13.1.1 FAQ: General

In this section TA's and students can add FAQ's from the piazza. As the material especially the programming related one is so useful that it is shared by now in multiple classes, however they use different piazza's sharing the information in an FAQ in the handbook allows us to quickly disseminate the relevant information between classes. If an FAQ is only for one class we will be especially mark it.

13.1.1.1 Can I assume that all information is in the FAQ to do the class?

No. The class book will be our main source of information not just a collection of FAQ's.

13.1.1.2 Piazza

13.1.1.2.1 Why are some FAQs that are on piazza not here?

Two reason:

1. some of them need not to be in this FAQ.
2. The TAs will evaluate the FAQs every day at the end of the day and integrate those that need to be in this list at that time. Hence it may take up to 24 hours for FAQs to appear here.

Once an FAQ is in the book answered (it may actually be part of another section, TA's will mark the FAQ in piazza, so you can make sure which FAQs are already in the book. We recommend to look in the book as there could be information in it that you otherwise missed.

13.1.1.3 How do I find all FAQ's in Piazza?

Two ways exist

- a. Please visit your class piazza. You will find a “faq” tag in your piazza window. Click on it and all posts marked with FAQ will show up,
- b. In the search field type in FAQ. All posts with the text FAQ in it will be listed.

13.1.1.4 Has SOIC computers I can use remotely?

See: <https://uisapp2.iu.edu/confluence-prd/pages/viewpage.action?pageId=114491559>

13.1.1.5 When contributing to the book my name is not listed

The following reasons exist:

1. if its not listed at all your contribution may be in a different repository, please contact Gregor
2. if it does not show up correctly and only shows your github name, which you can see in the contributor section or with

```
$ git shortlog -s -e
2 laszewsk <laszewski@gmail.com>
...
```

You need to do two things.

First, add your name to the file

- <https://github.com/cloudmesh-community/book/blob/master/.mailmap>

Second, complete the set up your git on the machine you work with in case you use a commandline tool with git init (see our notes on this)

If you use the GUI you may need to go to the account settings and associate a first name lastname, I however do not know ho to do that, so if you kwon reply ti


this

13.1.1.6 How to read the technical sections of the lecture notes

We will add throughout the semester some technical lecture notes. These notes contain information on how to install and run certain programs on a computer. What we have seen in the past with some students is that they do not read the text between the sections. Instead they just execute things without reading or understanding assuming that they can just copy and paste. These sections include valuable information that you **must** read before you execute any code in them.

Here is the workflow on how to read such technical sections

1. Do not execute anything yet
2. Read the entire section including the lines between the gray boxes
3. Step back and reflect on what you read
4. Reread the section, if a section needs more information google for it (things could be overnight updated on the internet, please remember we are just presenting a snapshot in time here)
5. Once you have obtained knowledge, decide if the section is relevant for you (e.g. windows sections may not be relevant for MacOS users)
6. Carefully execute the relevant portions for you

 *AS ALWAYS THERE IS NO GUARANTEE THAT WHAT THE CODE WORKS OR COULD NOT DESTROY SOMETHING. MAKE SURE TO HAVE A BACKUP. IF IN DOUBT RUN IN A VIRTUAL MACHINE IF YOU CAN.*

13.1.1.7 How to check if a yaml file is valid?

In case you need to check an open source public YAML file you can use the

following

The easiest is to use yamllint:

```
$ pip install yamllint
$ yamllint README.yml
```

Using yamllint is our preferred method.

A python script to check it is available at

- https://github.com/cloudmesh-community/book/tree/master/examples/yaml-validation/validate_yaml.py

This python script depends on `ruamel.yaml` package. We can install it using following command:

```
$ pip install ruamel.yaml
```

It accepts file path as an argument. This script will load `YAML` file and dump its content on console. For invalid syntax it will throw an error.

To execute python script you need to run following command after you clone *book* repository.

```
$ cd examples/yaml-validation
$ chmod +x validate_yaml.py
$ ./validate_yaml.py <path to yaml file to validate>
```

Online checkers are available at

- <https://codebeautify.org/yaml-validator>

A ruby script can do

```
$ ruby-e "require 'yaml';puts YAML.load_file('./README.yml)'"
```

YAML validation in visual studio can be achieved also

- <https://marketplace.visualstudio.com/items?itemName=redhat.vscode-yaml>

13.1.1.8 Download the ePub frequently

Please be reminded that the ePub is updated frequently and we recommend that you download it before you read.

I myself have integrated an ePub reader in my Web browser so that every time I click on the View Raw in github, I get the most up to date version.

I use ibooks on OSX, calibre is a good system on Windows and Linux, MS also has Microsoft Edge. However on Microsoft edge you will need the latest version which starts with 42

13.1.1.9 Spelling of filenames in github

Most of our scripts require proper spelling including proper capitalization. The spelling of `notebook.md` is `notebook.md`

not

`Notebook.md` or `NOTEBOOK.md` or other spelling

Please, correct if you did not use lower case

The spelling of `README.yaml` is `README.yaml` and not

`README.md` (which needs to be removed) or `readme.yaml`

please correct if needed. We will not grade any assignments if your README.yaml or notebook.md is misspelled or missing or is not following our simple format.

13.1.1.10 How to open the ePub from Github?

If you see the `View Raw`, you need to click on it. It will download the file. Then you can open that.

However, If you use edge or integrated your ePub viewer in your browser and clicking on it will automatically open your ePub browser.

13.1.1.11 Assignment Summary

🕒 outdated

- a. The assignment is discussed in Chapter 1 of the lecture notes
- b. Examples of what other students have done are in the Example Artifacts section

Please look at both sections

In this class we addressed 3 assignment that related to your grade

Tech summaries - they have been assigned to you in <https://piazza.com/class/jl6rxey6w413gi?cid=89> to show to the TAs that you work on them use the nomenclature that is discussed in the preface of the technology handbook. Put yours hid in the “headline” and a smiley when done, If you work on it put in a hand. Project - look at examples in the example artifact sections A paper has typically the following sections

Theory Implementation (e.g. Python) Benchmark

A more detailed outline is

- Paper
- Title
- Abstract
- Introductions
- Requirements
- Architecture
- Implementation
- Benchmark
- Conclusions
- Bibliography
- Work breakdown

13.1.1.12 Auto 80 char

🕒 outdated

Those that use emcas could experiment with the following. I do not know if this works well yet.

The following will autoformat an entire file to 80 chars. The reason i put it in test.md is that I do not know if it reliably works on all md files, just inspect the output and decide for yourself. some md files you may not want to manipulate with this though

```
cp file.md test.md
emacs -batch test.md --eval '(fill-region (point-min) (point-max))' -f save-buffer
```

13.1.1.13 Useful FAQs for residential and online students

🕒 this is outdated.

You will know if this post applies to you.

This class does not have a high volume on posts via Piazza

What we find is that some students create a high overhead on themselves by not following our FAQs or documentation on the technology summaries. When we observe something we just post it in an FAQ in piazza that we expect you look at. Yet we find some students that keep on resubmitting their technology summaries while not integrating our tips from the FAQs that cost less then a minute to do. Those that do not read and follow the FAQ make their work unnecessary complicated. We even start now noticing students that remove bibliography entries instead of just fixing them. Also, we saw recently students that had perfectly geat entries, other than the authors (see FAQ) and instead of fixing the authors in case it was a company or organization, to fix random fields such as the titles and thus creating even more work on themselves.

We have lots of online hours during the week There are 4 hours you can attend, Mo, We Thu, so if you do have something you do not understand, I recommend that you use these hours. In case you are a residential student you also have Fridays. To start, I would review our FAQs

Interestingly we see these issues more with residential students than with online students. This may indicate that the residential students in question forget to read the posts in piazza?

13.1.1.14 What if i committed a wrong file to github, a.g. a private key?

The answer to this question is more complicated than you think. Thus the best way to deal with it is to

AVOID IT:

- a. first do github adds file by fill with git add. Avoid using adds on AND DO NOT USE

```
git add . # <<<<<<< DO NOT USE
```

- b. only use ssh keys in ~/.ssh **NEVER** place keys in directories that are managed by git

YOU CAN NOT EASILY DELETE FILES FROM GIT:

- c. as you may already know despite you deleting a file from git it is still in the git history. Also there are bad characters out there so if you checked in you ssh private key just for a second

you must assume your private key is now compromised and all machines that use it are compromised.

- d. although Git allows you to delete the file, it is still in the githistory, which can be mined so despite you pressing delete its still there and can be found. This is not a bug in git but this is you having git not used right.
- e. There are ways to purge such files, but it would imply that everyone that did a fork needs to do a new fork which is naturally a big issue, so we do not do this during the semester.

NOW WHAT?

- f. every machine on which you used the public key of this private key is to be considered now compromised.
- g. put them off from the network while plugging out the network plug
- h. if the machines are not owned by you but for example, IU, notify the people

that own the machine to ask for help with mitigation.

- i. if you are lucky, replace the key, this is the case for example for services such as github. Make sure to inspect the configurations and see if your account has not been hijacked.
- j. We will immediately remove you from services such as future systems and chameleon cloud as a precaution or deactivate your membership in our cloud accounts.
- k. if you used the keys on other services, including IU, it is up to you to identify how to deal with this,
- l. definitely create a new key and use that from now on.
- m. you can call Gregors office number or use piazza to set up a call to identify what the impact is as this is typically an emergency use 812 856 1311. Do not leave a msg, but instead send e-mail.with your phone number so we can call you back to assess the situation.
- n. if you use them on public clouds that cost money, shut down all machines that use them. I would not start them again but instead use new once. It may be time to drop everything and do this first. Sorry for making you now panic.

13.1.2 FAQ: 516

This section contains FAQs relevant for 516. faq

13.1.2.1 Why have the lectures been removed from the resources section?

We identified that someone after almost three weeks has not looked at the ePub and missed valuable information. To force students to look at the epub we have removed the direct links to the md files in piazza Resources section. Instead we still leave the links to the md file sin the syllabus so that you can still get to them from the syllabus. This way also room for error is minimized as we only have to update one file wit the location and not two.

The little clouds in the ePub are automatically created and provide a direct link to the md file for which the content follows.

Essentially instead of making it real convenient not to read the ePub, we make it now real convenient that you read it while *hiding* the inks a bit in the syllabus md file.

13.1.2.2 Opening ePub properly in Linux

We recently noticed that not all the symbols, which are pretty important, are visible when we open the ePub in Linux using Calibre.

Before trying to putting time on fixing the Calibre, I tried couple of other software and seems like Bookworm can open the ePub correctly. I tried this in Ubuntu 18.04 (unity). You can follow the instructions here for installing Bookworm.

13.1.2.3 Microsoft Edge for ePub

On Windows 10 you need to have version 42.17134 for edge to properly display an ePub.

13.1.2.4 Project format: Markdown allowed

I have been working with some students on using Markdown for the format of the project report and it turns out to be possible to write your project report in markdown.

Hence, you have the option to use Markdown or LaTeX. We still use bibtex for management of the references in markdown.

tips for use of markdown for project report writing:

```
labels for citation in markdown are [@label]
```

use of commands

```
```bash
$ ls
```
```


use of python

```
```python
print("hello")
```
```

do not use quotes when you ought to use italics: This technology is called “docker” is wrong, but this technology is called *docker* is right. Use # to manage your sections not underscores or = this way you can easier count if indentation level is right There must only be the title with one # images are included as documented in our notation.md file of the lecture notes Do not use the words below and above, as the images can float and below and above makes ABSOLUTELY NO SENSE when referring to images if you do not know the final format where the images will be placed which is the case for us. Figures must be mentioned in the text explicitly. e.g. as shown in `@fig:figlabel` you can use pandoc locally to make sure bibs and figures show up corretcly.

13.1.2.5 pull request in documents from commandline

I updated a section in the github section called

“25.5.17 Contributing to the Document”

The problem is neither the TAs or I can test it as our permissions are different. Would someone with github knowledge be so kind to test that section and improve so others can replicate this?

It may actually work, but I have not tested it. Improvement suggestions are welcome.

There is lots of documentation about this on the github.com web page also,

If you are forking a repo, it’s better to keep your master branch clean. Meaning that there won’t be any changes done on your master branch itself. It will be the branch that you sync with the upstream repo (the cloudmesh repo).

Hence, if you are doing any changes, it’s better to do it in a branch in your own repo. Then if there are conflicts, you can resolve them on your own repo first when rebasing (syncing your dev branch with the upstream master).

Ex:

Let us say you are working on the file ref.bib.

You are keeping the master of your repo in sync and working on a branch called dev-ref. But when you try to push, you find out that there's already a new change in the upstream master for the same file.

Then what you do is, first sync your master. Then you do a rebase for your branch to get the branch up to date with the latest changes (your synced master). Once rebased (fixing the conflicts), then you can send the pull request from your branch to the upstream repo.

It's always better to keep your master clean for the rebase to be easier on your fork.

13.1.2.6 pyenv installed but can not find python

We recommend that you read the section that comes immediately after the pyenv install section. It comments on how to install python with pyenv. pyenv by itself does not come with python, it is a tool to install python. You naturally have to install the version you want to use.

13.1.2.7 ssh add/keychain on OSX

There is lots of information in the lecture notes on SSH I strongly recommend you read it. However, I saw some people struggle with SSH keychain on OSX. Here are some tips on how to make it work on OSX which I found on the Web.

1. Start the ssh-agent in the background.

```
eval "$(ssh-agent -s)"  
Agent pid 59566
```

2. Use `~/.ssh/config` file to automatically load keys into the ssh-agent and store passphrases in your keychain. Add the following to the file:

```
Host *  
  AddKeysToAgent yes  
  UseKeychain yes  
  IdentityFile ~/.ssh/id_rsa
```

3. Add your SSH private key to the ssh-agent and store your passphrase in the keychain. If you created your key with a different name, or if you are adding an existing key that has a different name, replace id_rsa in the command with the name of your private key file. Note the -K is OSX specific and means you will use keychain to manage your key

```
ssh-add -K ~/.ssh/id_rsa
```

13.1.2.8 Check into github early

There is no such thing as to check your things in github early. In fact that is good as than others can contribute early including me if I have time. A good example is the vagrant code where a student made the wise decision to check it in and someone (in this case me) contributed. As you can see through teamwork we can achieve a lot.

So do not be embarrassed if your code needs to have improvements

In the cm folder when you work on things just create an incoming directory and put your code in a subfolder if it is not yet integrated. We will set up a library structure soon.

I just spoke to a student who thought his code was not yet good enough. But my opinion is any code is good enough. as it allows others to contribute early

13.1.2.9 I want to start my project now

We have provided you with a concrete assignment of developing a python program. This program, can be used as part of you project. I advise you that if you like to start on your project now. TO start with the development of this program. We will enhance this program so that the project can benefit from it.

We have provided you with a concrete assignment of developing a python program. This program, can be used as part of you project. I advise you that if you like to start on your project now. TO start with the development of this program. We will enhance this program so that the project can benefit from it.

You should spend your time learning about python and setting up your

environment. Using VirtualBox will come in handy when we start managing virtual machines from the command line.

13.1.2.10 Mac with external hard drive

Apologies if this is explained in the book but I'm not very familiar with this, would using a macbook Air with an external 1TB hard drive be sufficient computing power/memory for this course or do I need to look into memory chips?

Yes. typically you do not need an external HD if you have space left. Naturally if your Mac book is 10 years old there are limitations. Certainly you can login into the cloud and use cloud resources from your mac. We even had student s that successfully conducted the class with a \$35 Raspberry pi. We know that chroembook, cell phone and tablet is not good enough.

13.1.2.11 Submitting a section

Do you want us to email or post here to let you know when we've created a section and a pull request for it? And do we need to check with you that no one else is doing that section before we write and submit it?

we wtch the pull requests actively in github, when you have a section taht is not yet in git hub add them and put in the comment you work on it,

13.1.2.12 Where to post comprehensions?

Where would you like us to post comprehensions? Will there be a thread for each comprehension or should we be making our own posts for each?

- a. When students have questions and others answer that counts as comprehension.
- b. Once we have set up the github repos you can post comprehension code you like to share with others there
- c. Residential students will have opportunity to also participate actively in the

class.

- d. There will be a notebook.md in which you can document your activities. That however will not be set up till Friday when the deadline for the survey is passed.

13.1.2.13 Communication of sections into the README.yml

see: <https://github.com/cloudmesh-community/sp19-516-122> as example

MAKE SURE YOU USE EXACTLY THE SAME SPACING AND DASHES AND ATTRIBUTES

Look at it in raw, only copy from the raw file, never from the rendered version

13.1.2.14 Question on adding new sections to the book and github protocol

I was interested in adding the following item below to the book in the data center section “6.4 Data Center Characteristics” as another bullet example of AWS failures that have occurred:
<https://www.datacenterknowledge.com/archives/2017/03/02/aws-outage-that-broke-the-internet-caused-by-mistyped-command>

I am new to github, what is the protocol to obtain a local copy of the book, update, then submit my change?

The easiest way the markdown file exists is to locate it in github via the Web browser and edit it in the browser.

Make the modification and create a pull request. We talked about this in the online and residential hours this week.

You can setup an online help with a TA to have them show you.

13.1.2.15 Where to post references

Is there a particular bib file we should be adding to for our references or should we be creating a new one for the sections/chapters we do?

first identify if the bib not already exists.

if not simplui add a new bibfile with the topic and add it there.

13.1.2.16 where are the lectures?

We got the first question that a student could not find the lectures, but had already access to piazza. In a previous post we send out a link to the ePub. This ePub is also linked in the Resources panel in piazza. You need to download and open this ePub.

As explained in that post on piazza you will need to install an ePub reader. On OSX this is installed by default and called iBooks or Books. On Windows you need to install a new version of Microsoft Edge and open the ePub for it or alternatively use Calibre. On Linux you need to install Calibre.

Every reasonable OS has working ePub readers ... if its not installed, please install one. We can discuss issues also live on Thursday if needed. Residential students can discuss issues on Friday morning in our first meeting.

The ePub contains a detailed set of information on how to take the class You need to read it., It also contains a section Proposed Lecture Timeline. I suggest you look at the first two entries in that table.

13.1.2.17 Swagger UI

For connexion based swagger projects, you can automatically enable the UI by requiring

```
connexion[swagger-ui]
```

instead of just

```
connexion
```

The UI will be available at `{apiurl}/ui`

13.1.2.18 Image file names (IMPORTANT)

as we simply copy all images from all papers into a single image directory, please make sure your filename is unique. Please add your hid as a prefix to your filename. Check the ePubs. If you see an issue simply rename the file and update the md file

13.1.2.19 Report template

A gentle reminder. I Need that most reread this as they are not following the instructions on how to integrate images

<https://github.com/cloudmesh-community/proceedings-fa18/blob/master/project-report/report.md>

There are some students that do it correctly though, so we know it works.

Be reminded that we use a more powerful markdown with figure labeling that github does not support.

We use pandoc with pandoc-fignos, and pandoc-citeproc. If you want to install it locally follow the instruction provided in the handbook. This way you can always check it locally before committing.

13.1.2.20 improved ePub vieweing of ePub documents

- - a. I installed a mor eup to date version of calibre
- - b. I embedded the emoji fonts
- - c. I updated the ePubs that show now emojis
- - d. ePub has also a TOC on the side.

As we had some users with outdated versions of MS Edge or a forwards button issue, we recommend that you give calibre a try, its relatively fast, however due to the size of the document you still ave to wait tilll the pa

13.1.2.21 empty lines in markdown

markdown is a simple format. One of the things it, however, does allow is that there are a variety of errors occurring if you use different markdown editors as they do not check for everything.

We HIGHLY RECOMMEND that you use ample empty lines.

- - a. there is an empty line before and after a heading
- - b. there is an empty line before and after a code block
- - c. there is an empty line before and after a quotation
- - d. there is an empty line before and after a figure
- - b. there is an empty line before and after a itemize list
- - b. there is an empty line before and after a paragraph (paragraphs are not indented)

...

so the rule is simple:

there is an empty line before and after each format change in markdown

So if your text does not display in the proceedings, correctly, I do recommend to follow this markdown recommendation.

13.1.2.22 Linux ePub Reader (followup)

Linux:

<https://itsfoss.com/best-ebook-readers-linux>

Windows:

<https://blog.kowalczyk.info/articles/epub-ebook-reader-viewer-for-windows.html>

<https://beebom.com/best-epub-reader-windows>

13.1.2.23 auto 80 char

Those that use emacs could experiment with the following. I do not know if this works well yet.

The following will autoformat an entire file to 80 chars. The reason i put it in test.md is that I do not know if it reliably works on all md files, just inspect the output and decide for yourself. some md files you may not want to manipulate with this though

```
cp file.md test.md
emacs -batch test.md --eval '(fill-region (point-min) (point-max))' -f save-buffer
```

13.1.2.24 what if i committed a wrong file to github, a.g. a private key?

AVOID IT:

- a. first do github adds file by fill with git add. Avoid using adds on AND DO NOT USE

```
git add . # <<<<<<< DO NOT USE
```

- b. only use ssh keys in ~/.ssh NEVER place keys in directories that are managed by git

YOU CAN NOT EASILY DELETE FILES FROM GIT:

- c. as you may already know despite you deleting a file from git it is still in the git history. Also there are bad characters out there so if you checked in you ssh private key just for a second

you must assume your private key is now compromised and all machines that use it are compromised.

- d. although Git allows you to delete the file, it is still in the githistory, which

can be mined so despite you pressing delete its still there and can be found. This is not a bug in git but this is you having git not used right.

- e. There are ways to purge such files, but it would imply that everyone that did a fork needs to do a new fork which is naturally a big issue, so we do not do this during the semester.

NOW WHAT?

- f. every machine on which you used the public key of this private key is to be considered now compromised.
- g. put them off from the network while plugging out the network plug
- h. if the machines are not owned by you but for example, IU, notify the people that own the machine to ask for help with mitigation.
- i. if you are lucky, replace the key, this is the case for example for services such as github. Make sure to inspect the configurations and see if your account has not been hijacked.
- j. We will immediately remove you from services such as future systems and chameleon cloud

as a precaution or deactivate your membership in our cloud accounts.

- i. if you used the keys on other services, including IU, it is up to you to identify how to deal with this,
- j. definitely create a new key and use that from now on.
- k. you can call my office number or use piazza to set up a call to identify what the impact is

as this is typically an emergency use 812 856 1311. Do not leave a msg, but instead send e-mail.with your phone number so we can call you back to assess the situation.

- l. if you use them on public clouds that cost money, shut down all machines

that use them. I would not start them again but instead use new ones. It may be time to drop everything and do this first. Sorry for making you now panic.

13.1.2.25 markdown and bibtex

As you know we do not use LaTeX for this class but simply markdown. you can use pandoc to create your ePubs locally if you wish while following the paradox manual.

However, it's much simpler than that, as we create the proceedings with all your markdown papers for you once a week, so you can check it. Often we create it multiple times a day.

So you do not have to do much more than once in a while looking at the ePubs.

As part of this we like to remind you that we did distribute on day one of the class a document located in

<https://github.com/cloudmesh-community/book/blob/master/README.md>

That is called Scientific Writing II. This includes a section about LaTeX which you can ignore, but it also includes a section about bibtex and how to do bibtex entries that you may be benefitting from. So take a quick look at the Section 3. It also explains how to improve bibtex entries which is important for your projects.

<http://cyberaide.org/papers/vonLaszewski-latex.pdf>

Also, we noticed that some do not follow our tips posed as part of the FAQ's we send out here.

So we do recommend that you inspect them. TAs are assigned to also move the FAQs into the handbook. So you can also find them there (after a week).

Please also be reminded that there is an empty line before and after a heading or a quote or

a list or any paragraph. paragraphs are not indented with tabs or spaces

13.2 AMAZON WEB SERVICE PRODUCTS

13.2.1 Compute

- [Amazon EC2](#)
- [Amazon EC2 Auto Scaling](#)
- [Amazon Elastic Container Service](#)
- [Amazon Elastic Container Service for Kubernetes](#)
- [Amazon Elastic Container Registry](#)
- [Amazon Lightsail](#)
- [AWS Batch](#)
- [AWS Elastic Beanstalk](#)
- [AWS Fargate](#)
- [AWS Lambda](#)
- [AWS Serverless Application Repository](#)
- [Elastic Load Balancing](#)
- [VMware Cloud on AWS](#)
- [AWS Marketplace](#)

13.2.2 Storage

- [Amazon Simple Storage Service \(S3\)](#)
- [Amazon Elastic Block Store \(EBS\)](#)
- [Amazon Elastic File System \(EFS\)](#)
- [Amazon Glacier](#)
- [AWS Storage Gateway](#)
- [AWS Snowball](#)
- [AWS Snowball Edge](#)
- [AWS Snowmobile](#)
- [AWS Marketplace](#)

13.2.3 Databases

- [Amazon Aurora](#)
- [Amazon RDS](#)
- [Amazon DynamoDB](#)
- [Amazon ElastiCache](#)

- [Amazon Redshift](#)
- [Amazon Neptune](#)
- [AWS Database Migration Service](#)
- [AWS Marketplace](#)

13.2.4 Migration

- [AWS Migration Hub](#)
- [AWS Application Discovery Service](#)
- [AWS Database Migration Service](#)
- [AWS Server Migration Service](#)
- [AWS Snowball](#)
- [AWS Snowball Edge](#)
- [AWS Snowmobile](#)
- [AWS Marketplace](#)

13.2.5 Networking & Content Delivery

- [Amazon VPC](#)
- [Amazon VPC PrivateLink](#)
- [Amazon CloudFront](#)
- [Amazon Route 53](#)
- [Amazon API Gateway](#)
- [AWS Direct Connect](#)
- [Elastic Load Balancing](#)
- [AWS Marketplace](#)

13.2.6 Developer Tools

- [AWS CodeStar](#)
- [AWS CodeCommit](#)
- [AWS CodeBuild](#)
- [AWS CodeDeploy](#)
- [AWS CodePipeline](#)
- [AWS Cloud9](#)
- [AWS X-Ray](#)
- [AWS Marketplace](#)

- [AWS Command Line Interface](#)
- [AWS Tools & SDKs](#)

13.2.7 Management Tools

- [Amazon CloudWatch](#)
- [AWS Auto Scaling](#)
- [AWS CloudFormation](#)
- [AWS CloudTrail](#)
- [AWS Config](#)
- [AWS OpsWorks](#)
- [AWS Service Catalog](#)
- [AWS Systems Manager](#)
- [AWS Trusted Advisor](#)
- [AWS Personal Health Dashboard](#)
- [AWS Command Line Interface](#)
- [AWS Management Console](#)
- [AWS Managed Services](#)
- [AWS Marketplace](#)

13.2.8 Media Services

- [Amazon Elastic Transcoder](#)
- [Amazon Kinesis Video Streams](#)
- [AWS Elemental MediaConvert](#)
- [AWS Elemental MediaLive](#)
- [AWS Elemental MediaPackage](#)
- [AWS Elemental MediaStore](#)
- [AWS Elemental MediaTailor](#)

13.2.9 Security, Identity & Compliance

- [AWS Identity and Access Management \(IAM\)](#)
- [Amazon Cloud Directory](#)
- [Amazon Cognito](#)
- [Amazon GuardDuty](#)
- [Amazon Inspector](#)

- [Amazon Macie](#)
- [AWS Certificate Manager](#)
- [AWS CloudHSM](#)
- [AWS Directory Service](#)
- [AWS Firewall Manager](#)
- [AWS Key Management Service](#)
- [AWS Organizations](#)
- [AWS Secrets Manager](#)
- [AWS Single Sign-On](#)
- [AWS Shield](#)
- [AWS WAF](#)
- [AWS Artifact](#)
- [AWS Marketplace](#)

13.2.10 Machine Learning

- [Amazon SageMaker](#)
- [Amazon Comprehend](#)
- [Amazon Lex](#)
- [Amazon Polly](#)
- [Amazon Rekognition](#)
- [Amazon Machine Learning](#)
- [Amazon Translate](#)
- [Amazon Transcribe](#)
- [AWS DeepLens](#)
- [Apache MXNet on AWS](#)
- [TensorFlow on AWS](#)
- [AWS Deep Learning AMIs](#)

13.2.11 Analytics

- [Amazon Athena](#)
- [Amazon EMR](#)
- [Amazon CloudSearch](#)
- [Amazon Elasticsearch Service](#)
- [Amazon Kinesis](#)
- [Amazon Redshift](#)

- [Amazon QuickSight](#)
- [AWS Data Pipeline](#)
- [AWS Glue](#)
- [AWS Marketplace](#)

13.2.12 Mobile

- [AWS Mobile Hub](#)
- [Amazon API Gateway](#)
- [Amazon Pinpoint](#)
- [AWS AppSync](#)
- [AWS Device Farm](#)
- [AWS Mobile SDK](#)

13.2.13 AR & VR

- [Amazon Sumerian](#)

13.2.14 Application Integration

- [Amazon MQ](#)
- [Amazon Simple Queue Service \(SQS\)](#)
- [Amazon Simple Notification Service \(SNS\)](#)
- [AWS AppSync](#)
- [AWS Step Functions](#)

13.2.15 Customer Engagement

- [Amazon Connect](#)
- [Amazon Pinpoint](#)
- [Amazon Simple Email Service \(SES\)](#)
- [AWS Marketplace](#)

13.2.16 Business Productivity

- [Alexa for Business](#)
- [Amazon WorkDocs](#)

- [Amazon WorkMail](#)
- [Amazon Chime](#)

13.2.17 Desktop & App Streaming

- [Amazon WorkSpaces](#)
- [Amazon AppStream 2.0](#)

13.2.18 Internet of Things

- [AWS IoT Core](#)
- [Amazon FreeRTOS](#)
- [AWS Greengrass](#)
- [AWS IoT 1-Click](#)
- [AWS IoT Analytics](#)
- [AWS IoT Button](#)
- [AWS IoT Device Defender](#)
- [AWS IoT Device Management](#)

13.2.19 Game Development

- [Amazon GameLift](#)
- [Amazon Lumberyard](#)

13.2.20 AWS Marketplace Software

- [Infrastructure Software \(1300+\)](#)
- [Business Software \(845+\)](#)
- [Developer Tools \(220+\)](#)

13.2.21 AWS Cost Management

- [AWS Cost Explorer](#)
- [AWS Budgets](#)
- [Reserved Instance Reporting](#)
- [AWS Cost and Usage Report](#)

13.2.22 Exercise

E.AWS.0:

Identify all products related to IaaS service offerings and mark them with `<code>#</code>` after the bullet. Do copy the `aws.md` file into your own repository, do not yet create a pull request. Confirm in a team your findings and agree with each other.

13.3 MICROSOFT AZURE AND CLOUD PRODUCTS

Just as Amazon Microsoft offers a large number of services. Many of them are not just IaaS services. We list here the from Microsoft featured services as of Sep. 2018. The ones marked with a `<code>#</code>` are related to IaaS.

The services are organized in the following categories:

- AI + Machine Learning
- Analytics
- Compute
- Containers
- Databases
- Developer Tools
- DevOps
- Identity
- Integration
- Internet of Things
- Management Tools
- Media
- Migration
- Mobile
- Networking
- Security
- Storage
- Web

The following subsections are obtained while looking at the Web page and

copying the “key phrases”. In a future version we will improve the section while curating the service descriptions.

13.3.1 AI + Machine Learning

Source: <https://support.microsoft.com/en-us/allproducts>

[AI + Machine Learning](#)- Create applications using artificial intelligence capabilities

- [Cognitive Services](#) - Add smart API capabilities to enable contextual interactions
- [Azure Bot Service](#) - Intelligent, serverless bot service that scales on demand
- [Azure Databricks](#) - Fast, easy, and collaborative Apache Spark-based analytics platform
- [Machine Learning](#) - Open and elastic AI development spanning the cloud and the edge
- [Cognitive Services Search APIs](#) - Harness the ability to comb billions of webpages, images, videos, and news with a single API call
- [Cognitive Services - Language APIs](#) - Process natural language with pre-built scripts, evaluate sentiment, and learn to recognize intent
- [Cognitive Services - Vision APIs](#) - Use Image-processing algorithms to smartly identify, caption and moderate your pictures
- [Cognitive Services - Speech APIs](#) - Convert speech to text or text to speech, translate text or audio, or add speaker recognition to your app
- [Cognitive Services - Knowledge APIs](#) - Map information and data in order to solve complex tasks
- [See more](#)

13.3.2 Analytics

Source: <https://support.microsoft.com/en-us/allproducts>

[Analytics](#) - Gather, store, process, analyze, and visualize data of any variety, volume, or velocity

- [SQL Data Warehouse](#) - Elastic data warehouse as a service with enterprise-class features Azure Databricks - Fast, easy, and collaborative Apache

- Spark-based analytics platform
- [HDInsight](#) - Provision cloud Hadoop, Spark, R Server, HBase, and Storm clusters
 - [Data Factory](#) - Hybrid data integration at enterprise scale, made easy
 - [Machine Learning](#) - Open and elastic AI development spanning the cloud and the edge
 - [Stream Analytics](#) - Real-time data stream processing from millions of IoT devices
 - [Data Lake Analytics](#) - Distributed analytics service that makes big data easy
 - [Azure Analysis Services](#) - Enterprise-grade analytics engine as a service
 - [Event Hubs](#) - Receive telemetry from millions of devices
 - [See more](#)

13.3.3 Compute

Source: <https://support.microsoft.com/en-us/allproducts>

Core IaaS [Compute](#) access to cloud resources.

- [Virtual Machines](#) - Provision Windows and Linux virtual machines in seconds
- [Virtual Machine Scale Sets](#) - Manage and scale up to thousands of Linux and Windows virtual machines
- [Azure Kubernetes Service \(AKS\)](#)
- [Functions](#) - Process events with serverless code
- [Service Fabric](#) - Develop microservices and orchestrate containers on Windows or Linux
- [App Service](#) - Quickly create powerful cloud apps for web and mobile
- [Container Instances](#) - Easily run containers on Azure without managing servers
- [Batch](#) - Cloud-scale job scheduling and compute management
- [Azure Batch AI](#) - Easily experiment and train your deep learning and AI models in parallel at scale
- [See more](#)

13.3.4 Containers

Source: <https://support.microsoft.com/en-us/allproducts>

[Containers](#) - Develop and manage your containerized applications faster with integrated tools

- [Azure Kubernetes Service \(AKS\)](#)
- [Container Instances](#) - Easily run containers on Azure without managing servers
- [Service Fabric](#) - Develop microservices and orchestrate containers on Windows or Linux
- [Container Registry](#) - Store and manage container images across all types of Azure deployments
- [App Service](#) - Quickly create powerful cloud apps for web and mobile
- [Web App for Containers](#) - Easily deploy and run containerized web apps that scale with your business
- [Batch](#) - Cloud-scale job scheduling and compute management
- [See more](#)

13.3.5 Databases

Source: <https://support.microsoft.com/en-us/allproducts>

[Databases](#) - Support rapid growth and innovate faster with secure, enterprise-grade, and fully managed database services

- [Azure Cosmos DB](#) - Globally distributed, multi-model database for any scale
- [Azure SQL Database](#) - Managed relational SQL Database as a service
- [Azure Database for MySQL](#) - Managed MySQL database service for app developers
- [Azure Database for PostgreSQL](#) - Managed PostgreSQL database service for app developers
- [SQL Server on Virtual Machines](#) - Host enterprise SQL Server apps in the cloud
- [SQL Data Warehouse](#) - Elastic data warehouse as a service with enterprise-class features
- [Azure Database Migration Service](#) - Simplify on-premises database migration to the cloud

- [Redis Cache](#) - Power applications with high-throughput, low-latency data access
- [SQL Server Stretch Database](#) - Dynamically stretch on-premises SQL Server databases to Azure
- [See more](#)

13.3.6 Developer Tools

Source: <https://support.microsoft.com/en-us/allproducts>

[Developer Tools](#)

- Build, manage, and continuously deliver cloud applications—using any platform or language
- [Visual Studio](#) - The powerful and flexible environment for developing applications in the cloud
- [Visual Studio Code](#) - A powerful, lightweight code editor for cloud development
- [SDKs](#) - Get the SDKs and command-line tools you need
- [Azure DevOps](#) - Services for teams to share code, track work, and ship software
- [CLIs](#) - Build, deploy, diagnose, and manage multi-platform, scalable apps and services
- [Azure Pipelines](#) - Continuously build, test, and deploy to any platform and cloud
- [Azure Lab Services](#) - Set up labs for classrooms, trials, development and testing, and other scenarios
- [Azure DevTest Labs](#) - Quickly create environments using reusable templates and artifacts
- [Developer tool integrations](#) - Use the development tools you know—including Eclipse, IntelliJ, and Maven—with Azure
- [See more](#)

13.3.7 DevOps

Source: <https://support.microsoft.com/en-us/allproducts>

DevOps - Deliver innovation faster with simple, reliable tools for continuous delivery

- [Azure DevOps](#) - Services for teams to share code, track work, and ship software
- [Azure Pipelines](#) - Continuously build, test, and deploy to any platform and cloud
- [Azure Boards](#) - Plan, track, and discuss work across your teams
- [Azure Repos](#) - Get unlimited, cloud-hosted private Git repos for your project
- [Azure Artifacts](#) - Create, host, and share packages with your team
- [Azure Test Plans](#) - Test and ship with confidence with a manual and exploratory testing toolkit
- [Azure DevTest Labs](#) - Quickly create environments using reusable templates and artifacts
- [DevOps tool integrations](#) - Use your favorite DevOps tools with Azure
- [Application Insights](#) - Detect, triage, and diagnose issues in your web apps and services
 - [See more](#)

13.3.8 Identity

Source: <https://support.microsoft.com/en-us/allproducts>

[Identity](#) - Manage user identities and access to protect against advanced threats across devices, data, apps, and infrastructure

- [Azure Active Directory](#) - Synchronize on-premises directories and enable single sign-on
 - [Azure Active Directory B2C](#) - Consumer identity and access management in the cloud
- [Azure Active Directory Domain Services](#) - Join Azure virtual machines to a domain without domain controllers
- [Azure Information Protection](#) - Better protect your sensitive information—anytime, anywhere
 - [See more](#)

13.3.9 Integration

Source: <https://support.microsoft.com/en-us/allproducts>

[Integration](#) - Seamlessly integrate on-premises and cloud-based applications, data, and processes across your enterprise

- [Logic Apps](#) - Automate the access and use of data across clouds without writing code
- [Service Bus](#) - Connect across private and public cloud environments
- [API Management](#) - Publish APIs to developers, partners, and employees securely and at scale
- [Event Grid](#) - Get reliable event delivery at massive scale
 - [See more](#)

13.3.10 Internet of Things

Source: <https://support.microsoft.com/en-us/allproducts>

[Internet of Things](#) - Bring IoT to any device and any platform, without changing your infrastructure

- [IoT Hub](#) - Connect, monitor and manage billions of IoT assets
- [IoT Edge](#) - Extend cloud intelligence and analytics to edge devices
- [IoT Central](#) - Experience the simplicity of SaaS for IoT, with no cloud expertise required
- [IoT solution accelerators](#) - Create fully customizable solutions with templates for common IoT scenarios
- [Azure Sphere](#) - Securely connect MCU-powered devices from the silicon to the cloud
- [Time Series Insights](#) - Explore and analyze time-series data from IoT devices
- [Azure Maps](#) - Simple and secure location APIs provide geospatial context to data
- [Functions](#) - Process events with serverless code
- [Event Grid](#) - Get reliable event delivery at massive scale
- [See more](#)

13.3.11 Management Tools

Source: <https://support.microsoft.com/en-us/allproducts>

[Management Tools](#) - Simplify, automate, and optimize the management of your Azure services

- [Microsoft Azure portal](#) - Build, manage, and monitor all Azure products in a single, unified console
- [Cloud Shell](#) - Streamline Azure administration with a browser-based shell
- [Azure Advisor](#) - Your personalized Azure best practices recommendation engine
- [Backup](#) - Simple and reliable server backup to the cloud
- [Cost Management](#) - Optimize what you spend on the cloud, while maximizing cloud potential
- [Azure Policy](#) - Implement corporate governance and standards at scale for Azure resources
- [Azure Monitor](#) - Highly granular and real-time monitoring data for any Azure resource
- [Log Analytics](#) - Collect, search, and visualize machine data from on-premises and cloud
- [Site Recovery](#) - Orchestrate protection and recovery of private clouds
- [See more](#)

13.3.12 Media

Source: <https://support.microsoft.com/en-us/allproducts>

Media - Deliver high-quality video content anywhere, any time, and on any device

- [Media Services](#) - Encode, store, and stream video and audio at scale
- [Encoding](#) - Studio grade encoding at cloud scale
- [Azure Media Player](#) - A single layer for all your playback needs
- [Live and On](#) -Demand Streaming - Deliver content to virtually all devices with scale to meet business needs
- [Media Analytics](#) - Uncover insights from video files with speech and vision services

- [Content Protection](#) - Securely deliver content using AES, PlayReady, Widevine, and Fairplay
- [Video Indexer](#) - Unlock video insights
- [See more](#)

13.3.13 Microsoft Azure Stack

Source: <https://support.microsoft.com/en-us/allproducts>

Microsoft Azure Stack is an extension of Azure—bringing the agility and innovation of cloud computing to your on-premises environment and enabling the only hybrid cloud that allows you to build and deploy hybrid applications anywhere. We bring together the best of the edge and cloud to deliver Azure services anywhere in your environment.

13.3.14 Migration

Source: <https://support.microsoft.com/en-us/allproducts>

[Migration](#)(<https://azure.microsoft.com/en>)(<https://azure.microsoft.com/en-us/services/azure-migrate/>) - Simplify and accelerate your migration to the cloud

- [Site Recovery](#) - Orchestrate protection and recovery of private clouds
- [Azure Database Migration Service](#) - Simplify on-premises database migration to the cloud
- [Data Box](#) - Secure, ruggedized appliance for Azure data transfer
- [Cost Management](#) - Optimize what you spend on the cloud, while maximizing cloud potential
- [See more](#) “Mobile Apps”)

13.3.15 Mobile

Source: <https://support.microsoft.com/en-us/allproducts>

[Mobile](#) - Build and deploy cross-platform and native apps for any mobile device

- [Mobile Apps - Build and host the backend for any mobile app-
[Notification Hubs](#) - Send push notifications to any platform from any back

end

- [Visual Studio App Center](#) - Ship apps faster by automating application lifecycles
- [Xamarin](#) - Create cloud-powered mobile apps faster
- [Azure Maps](#) - Simple and secure location APIs provide geospatial context to data
- [API Apps](#) - Easily build and consume Cloud APIs
- [See more](#)

13.3.16 Networking

Source: <https://support.microsoft.com/en-us/allproducts>

[Networking](#) - Connect cloud and on-premises infrastructure and services to provide your customers and users the best possible experience

- [Virtual Network](#) - Provision private networks, optionally connect to on-premises datacenters
- [Load Balancer](#) - Deliver high availability and network performance to your applications
- [Application Gateway](#) - Build secure, scalable, and highly available web front ends in Azure
- [VPN Gateway](#) - Establish secure, cross-premises connectivity
- [Azure DNS](#) - Host your DNS domain in Azure
- [Content Delivery Network](#) - Ensure secure, reliable content delivery with broad global reach
- [Azure DDoS Protection](#) - Protect your applications from Distributed Denial of Service (DDoS) attacks
- [Traffic Manager](#) - Route incoming traffic for high performance and availability

- [ExpressRoute](#) - Dedicated private network fiber connections to Azure

- [See more](#)

13.3.17 Security

Source: <https://support.microsoft.com/en-us/allproducts>

Protect your enterprise from advanced threats across hybrid cloud workloads

- [Security Center](#)
 - Unify security management and enable advanced threat protection across hybrid cloud workloads
 - [Key Vault](#) - Safeguard and maintain control of keys and other secrets
- [Application Gateway](#) - Build secure, scalable, and highly available web front ends in Azure
- [Azure Information Protection](#) - Better protect your sensitive information—anytime, anywhere
- [VPN Gateway](#) - Establish secure, cross-premises connectivity
- [Azure Active Directory](#) - Synchronize on-premises directories and enable single sign-on
- [Azure DDoS Protection](#) - Protect your applications from Distributed Denial of Service (DDoS) attacks
- [Azure Advanced Threat Protection](#) - Detect and investigate advanced attacks on-premises and in the cloud
- [See more](#)

13.3.18 Storage

Source: <https://support.microsoft.com/en-us/allproducts>

[Storage](#) - Get secure, massively scalable cloud storage for your data, apps, and workloads

- [Storage](#) - Durable, highly available, and massively scalable cloud storage
- [Blob Storage](#) - REST-based object storage for unstructured data
- [Archive Storage](#) - Industry leading price point for storing rarely accessed data
- [Queue Storage](#) - Effectively scale apps according to traffic
- [File Storage](#) - File shares that use the standard SMB 3.0 protocol
- [Disk Storage](#) - Persistent, secured disk options supporting virtual machines
- [Azure Data Lake Storage](#) - Massively scalable data lake storage
- [Data Box](#) - Secure, ruggedized appliance for Azure data transfer
- [Storage Explorer](#) - View and interact with Azure Storage resources
- [See more](<https://azure.microsoft.com/en>)(<https://azure.microsoft.com/en-us/services/#storage> “See more”)

13.3.19 Web

Source: <https://support.microsoft.com/en-us/allproducts>

Build, deploy, and scale powerful web applications quickly and efficiently

- [Web Apps](#) - Quickly create and deploy mission critical web apps at scale
- [API Management](#) - Publish APIs to developers, partners, and employees securely and at scale
- [Content Delivery Network](#) - Ensure secure, reliable content delivery with broad global reach
- [Azure Search](#) - Fully-managed search-as-a-service
- [Azure SignalR Service](#) - Add real-time web functionalities easily

13.4 FUTURESYSTEMS

This section gives an overview of the FutureSystems that are available as part of the DSC infrastructure. We cover the creation of FutureSystems Account, Uploading SSH Key and how to instantiate and log into Virtual Machine and accessing IPython are covered. In the end we discuss about running Python and Java on Virtual Machine.

13.4.1 FutureSystems evolved from FutureGrid

In this video we introduce FutureGrid a precursor to FutureSystems.



[Systems 12:12 FutureGid](#)

At this time we are replacing several of the older systems. To use these new systems you need to ask for access through them via our portal.

13.4.2 Creating Portal Account

This lesson explains how to create a portal account, which is the first step in gaining access to FutureSystems.

See Lesson 4 and 7 for SSH key generation on Linux, macOS or Windows.

 [Python 11:50 FutureGrid Introduction](#)

13.4.3 SSH Key Generation using ssh-keygen command

SSH keys are used to identify user accounts in most systems including FutureSystems. This lesson walks you through generating an SSH key via ssh-keygen command line tool.

 [Python 4:06 ssh-key gen](#)

13.4.4 Shell Access via SSH

This lesson explains how to get access FutureSystems resources vis SSH terminal with your registered SSH key.

 [Python 2:34 Shell Access via SSH](#)

13.4.5 Advanced SSH

This lesson shows you how to write SSH ‘config’ file in advanced settings.

 [Python 2:47 Advanced SSH](#)

13.4.6 SSH Key Generation via putty

This lesson is for Windows users. You will learn how to create an SSH key using PuTTYgen, add the public key to you FutureSystems portal, and then login using the PuTTY SSH client.

 [Python 3:51 Windows users](#)

13.4.7 FutureSystems Facilities

FutureSystems system resources are located at Indiana University (Bloomington). Resources at Indiana University (Bloomington) include a

1. 128-core HP cluster (Bravo),
2. 92-core cloud cluster (Echo),
3. 192-core Tesla GPU cluster (Delta),
4. 3456-core Haswell cluster (Juliet),
5. 126-core NVIDIA K80/Volta GPU cluster (Romeo),
6. 3264-core Knight's Landing cluster (Tango),
7. 480-core Platinum cluster (Tempest),
8. 768-core Platinum cluster (Victor).

Details of the resources are listed in subsequent sections.

13.4.7.1 Bravo

The large-memory HP cluster (Bravo) is a 1.7 Tflop HP Proliant distributed shared memory cluster with 128 processor cores and 3 TB total memory capacity. The compute nodes consist of 16 HP DL180 servers, each with two quad-core Intel Xeon E5620 2.4 GHz processors, 192 GB of memory, 12 TB of local attached storage, and a PCIe 4x QDR InfiniBand adapter for high bandwidth, low-latency MPI applications. Bravo is currently used as a shared storage cluster and is not being utilized for compute jobs. Operating System: RedHat Linux 6.9.

13.4.7.2 Delta

The GPU cluster (Delta) is a SuperMicro distributed shared memory cluster with 192 CPU cores and 14,336 GPU cores and 3TB total memory capacity. The compute nodes consist of 16 SuperMicro X8DTG-QF servers, each with 2 6-core Intel Xeon 5660 2.80 GHz processors, 2 nVIDIA Tesla C2075 GPUs with 448 cores per GPU, 192GB of memory, 9TB of local attached storage, and a Mellanox ConnectX-2 VPI dual-port InfiniBand QDR/10GigE PCIe adapter card. Operating System: RedHat Linux 7.4

13.4.7.3 Echo

The cloud cluster (Echo) is a SuperMicro distributed shared memory cluster with 192 CPU cores and 6TB total memory capacity. The compute nodes consist of 16 SuperMicro X9DRW servers, each node with 2 6-core Intel(R) Xeon(R) CPU E5-2640 2.50GHz processors; 384GB of memory, 10TB of local disk storage, a 10GbE Ethernet and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s onboard adapter for high bandwidth, low-latency MPI applications. Operating System: Ubuntu Linux 16.04

13.4.7.4 Juliet

The Haswell cluster (Juliet) is a SuperMicro distributed shared memory cluster with 3456 CPU cores and 16TB total memory capacity. The compute nodes consist of SuperMicro X10DRT-HIBF servers, 32 nodes with 2 18-core Intel(R) Xeon(R) CPU E5-2699 v3 2.30GHz processors; 96 nodes with 2 12-core Intel(R) Xeon(R) CPU E5-2670 v3 2.30GHz processors, all compute nodes with 128GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s onboard adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4

13.4.7.5 Romeo

The K80/Volta GPU cluster (Romeo) is a SuperMicro distributed shared memory cluster with 126 CPU cores, 161792 CUDA cores, and 768GB total memory capacity. The compute nodes consist of 4 SuperMicro X10DGQ servers with 2 12-core Intel(R) Xeon(R) CPU E5-2670 v3 2.30GHz processors, 4 NVIDIA GK210GL [Tesla K80] GPU Accelerator cards with 4992 CUDA cores, and 2 SuperMicro X10DGO servers with 2 10-core Intel Xeon E5-2600 v4 2.2GHz processors and 8 NVIDIA V100 (Tesla Volta) accelerators with 5120 CUDA cores. All nodes with 128GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s onboard adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4

13.4.7.6 Tango

The Knight's Landing cluster (Tango) is a Penguin Computing distributed

shared memory cluster with 3264 Xeon Phi cores and 12.8TB total memory capacity. The compute nodes consist of 16 nodes with 1 72-core Intel(R) Xeon Phi(TM) CPU 7290F 1.50GHz processor and 48 nodes with 1 68-core Intel(R) Xeon Phi(TM) CPU 7250F 1.50GHz processor. All nodes with 200GB of memory, 3.2TB of local disk storage, 800GB of NVMe storage, and an Intel OmniPath adapter for high bandwidth, low-latency MPI applications. Operating System: CentOS release 7.2.1511

13.4.7.7 Tempest

The Platinum cluster (Tempest) is a SuperMicro distributed shared memory cluster with 480 CPU cores and 2.5TB total memory capacity. The 10 compute nodes consist of SuperMicro X11DPT-PS servers with 2 24-core Intel(R) Xeon(R) Platinum 8160 2.10GHz processors, 256GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and an Intel OmniPath adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4

13.4.7.8 Victor

The Platinum cluster (Victor) is a SuperMicro distributed shared memory cluster with 768 CPU cores and 2.5TB total memory capacity. The 16 compute nodes consist of SuperMicro X11DPT-PS servers with 2 24-core Intel(R) Xeon(R) Platinum 8160 2.10GHz processors, 256GB of memory, 8TB of local disk storage, 400GB of NVMe storage, and a Mellanox ConnectX-3 InfiniBand FDR 56GT/s adapter for high bandwidth, low-latency MPI applications. Operating System: RedHat Linux 7.4 (see [Figure 67](#))

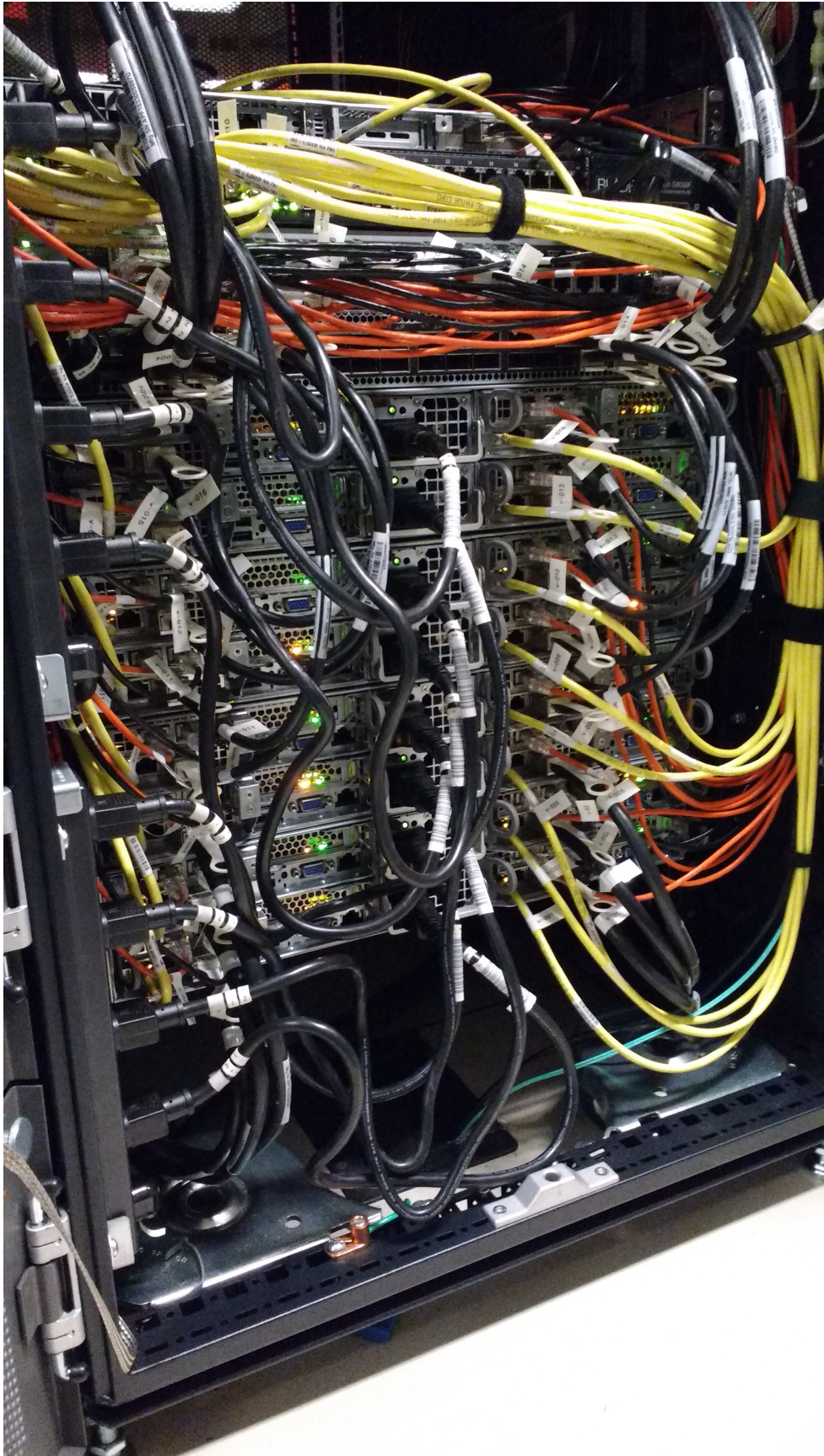


Figure 67: Cabling of Victor

13.4.7.9 PI Cluster

The details will be added once we have completed the purchase.

13.5 CHAMELEON

13.5.1 Resources

You can also visit the Chameleon Web page as there is more information about other topics that we may not need to worry about. This mostly includes using chameleon cloud as a bare metal resource and appliances, which we typically will not use.

If you prefer you can also go to the Chameleon Web site using the following links. However we have improved some of the documentation found in this document. The links to Chameleons online resources are:

- [Web page](#)
- [Documentation](#)
- [News](#)
- [About](#)
- [Log in](#)
- [Dashborad](#)

However, we have augmented some information and even provided corrections that are not covered on the Chameleon cloud Web Page. The information that overlap between the Web page and this material have bebeen copied and modified with permission from the Chameleoncloud team.

13.5.1.1 Outages

Any computer system may undergo maintenance. Before filing tickets with Chameleon cloud, make sure that the cloud is operational. Outages are posted at

<https://www.chameleoncloud.org/user/outages/>

13.5.1.2 Account Creation

The first step to get access Chameleon cloud is to create a user account if you do not already have one. You can skip to the next section if you have a chameleon cloud account.

The register web page is available at:

<https://www.chameleoncloud.org/user/register/>

For more details, please also consult the chameleon chapter in the handbook.

13.5.1.3 Join a Project

An active project is required to access compute resources. Each class has a particular project number that you will need to write down as you will use it to interact with the system. The information is given out by the instructor.

For the Fall 2019 516 related classes please use the following project number:

- [CH-819337](#)

However, before you can access it, the instructor (in our class Dr. von Laszewski) needs to authorize you to use the project. For this you have filled out an account survey that you were pointed to in piazza. The most common errors we see are that students provide us with the wrong user name or have not applied for a chameleon account. Once the instructor has added you, you will be able to use VM's on Chameleon cloud.

***Note:** If you use chameleon cloud for another class, please consult with your instructor and obtain a project number from them. You will have to let them know your chameleon user ID, so they can activate your account.*

13.5.1.4 Usage Restriction

As using VM's in a shared environment has an impact on resources for all, you are **REQUIRED** to shut down your resources after you are not using them anymore. Furthermore, before the class is over and we assign grades you must terminate your instances and free all IP addresses. Remember that any running VM is just like you were running a real computer. I am sure you close the lid of your laptop when not in use. Shutting down the VM is similar and avoids that you unnecessarily use resources that others could use in a shared environment. Furthermore, if you keep the machines running you will use energy and have an impact on the environment. **Be a team player** and be environmentally conscious by shutting down your non used instances. The way we use Chameleon Cloud is as a large team. As such please avoid deleting other people's VM's. All VMs that do not follow a particular naming scheme are subject to deletion at any time.

The pattern of the VMs must follow:

- any VM owned by gregor is allowed: `*gregor*` if it is owned by Gregor
- any VM with the name `NNN-firstname-i`, where NNN is the last digit of your ID from class, firstname is your first name and i is an integer.

13.5.2 Chameleon Cloud Hardware

The Chameleon architecture consists of a set of standard cloud units (SCUs), each of which is a single rack with 42 compute nodes, 4 storage nodes attached to 128TB of local storage in configurable arrays, and an OpenFlow compliant network switch. In addition to the homogeneous SCUs, a variety of heterogeneous hardware types is available to experiment with alternative technologies. The testbed also includes a shared infrastructure with a persistent storage system accessible across the testbed, a top-level network gateway to allow access to public networks, and a set of management and provisioning servers to support user access, control, monitoring and configuration of the testbed. Chameleon is physically distributed between the Texas Advanced Computing Center (TACC) and the University of Chicago (UC) through 100Gbps Internet2 links, to allow users to examine the effects of a distributed cloud.

Hardware Summary

Standard Cloud Units (SCUs)

Homogeneous Hardware Types

| | |
|------------------------------------|-------------------------|
| Number of Nodes per Rack: | |
| Local Storage per homogeneous SCU: | 128TB (configurable) |
| Network Switch: | OpenFlow Compliant |
| TACC/UC Distributed Cloud | 100Gbps Internet2 links |

As chameleon cloud updates their hardware we recommend to take a look at their hardware portal page. This page provides detailed information through their [Chameleon cloud Resource Discovery Portal](#)

Unfortunately, Chameleon Cloud only uses 42 (as of Sep 11 2019) of its nodes with Infiniband connections. Previous efforts such as FutureGrid were able to utilize all 128 nodes connected via Infiniband. However FutureGrid is no longer active. If you need better network performance you can use SDSC’s comet that provides virtual clusters while the nodes are connected all to Infiniband. However, comet does not use OpenStack, but provides a very convenient command line tool while leveraging cloudmesh.

13.5.2.1 Standard Cloud Units

The homogeneous standard cloud unit is a self-contained rack with all the components necessary to run a complete cloud infrastructure, and the capability to combine with other units to form a larger experiment. The rack consists of 42 Dell R630 servers; each with 24 cores delivered in dual socket Intel Xeon E5-2670 v3 “Haswell” processors (each with 12 cores @ 2.3GHz) and 128 GiB of RAM. In addition to the compute servers, each unit contains storage hosted in two FX2 chassis, each containing two Dell FC430 servers attached to two Dell PowerEdge FD332 storage blocks containing 16 2TB hard drives, for a total of 128TB of raw disk storage per unit. These FC430 storage nodes contain dual socket Intel Xeon E5-2650 v3 “Haswell” processors (each with 10 cores @ 2.3 GHz), 64 GiB of RAM, and can be combined across SCUs to create a Big Data infrastructure with more than a PB of storage. Each node in the SCU connects to a Dell switch at 10Gbps, with 40Gbps of bandwidth to the core network from each SCU. The total system contains 12 SCUs (10 at TACC and 2 at UC) for a total of 13,056 cores, 66 TiB of RAM, and 1.5PB of configurable storage in the SCU subsystem. (See [Figure 68](#))

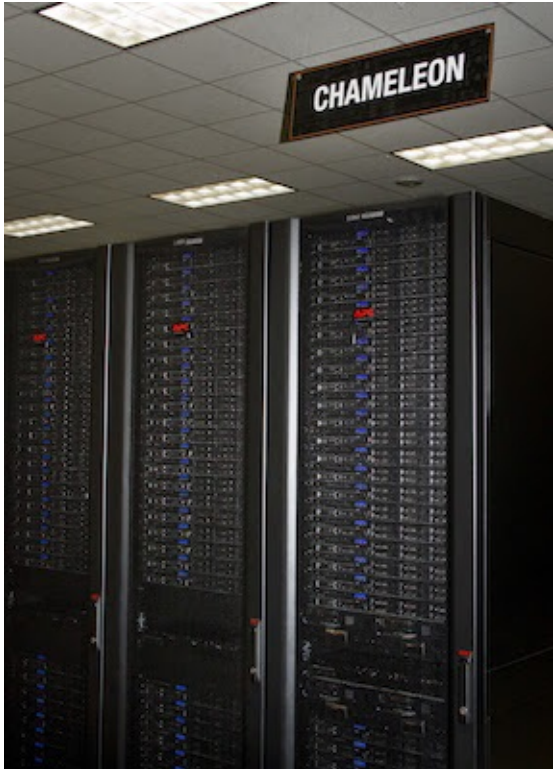


Figure 68: Chameleon Cloud Racks

13.5.2.2 Network

Networking is changing rapidly, and the network fabric is as much a part of the research focus of Chameleon as the compute or storage. For the Chameleon network, every switch in the research network is a fully OpenFlow compliant programmable Dell S6000-ON switch. Each node connects to this network at 10 Gbps, and each unit uplinks with 40Gbps per rack to the Chameleon core network. The core switches (Dell S6000-ON) are connected by 40 Gbps Ethernet links, which connect to the backbone 100Gbps services at both UC and TACC. A Fourteen Data Rate (FDR) Infiniband network (56Gbps) is also deployed on one SCU to allow exploration of alternate networks.

13.5.2.3 Shared Storage

While storage is dynamically provisioned to researchers to be used as an experiment needs within the SCUs, Chameleon also provides a shared storage system. The shared storage provides more than 3.6PB of raw disk in the initial configuration, which is partitioned between a file system and an object store that

is persistent between experiments. The shared storage is comprised of four Dell R630 servers with 128 GiB of RAM, four MD3260 external drive arrays, and six MD3060e drive expansion chassis, populated by 600 6TB near line SAS drives. The system also includes a dozen PowerEdge R630 servers as management nodes to provide for login access to the resource, data staging, system monitoring, and hosting various OpenStack services.

13.5.2.4 Heterogeneous Compute Hardware

The heterogeneous hardware includes various technologies: GPU and FPGA accelerators, SSD and NVMe storage, low-power ARM, Atom, and Xeon systems-on-a-chip. With the exception of the low-power systems-on-a-chip, each of the additional nodes is a Dell PowerEdge R730 server with the same CPUs as the R630 servers in the SCUs.

The two storage hierarchy nodes have been designed to enable experiments using multiple layers of caching: they are configured with 512 GiB of memory, two Intel P3700 NVMe of 2 TB each, four Intel S3610 SSDs of 1.6 TB each, and four 15K SAS HDDs of 600 GB each.

The GPU offering consists of two K80 GPU nodes, two M40 GPU nodes, sixteen P100 GPU nodes. These nodes target experiments using accelerators to improve the performance of some algorithms, experiments with new visualization systems, and deep machine learning. Each K80 GPU node is upgraded with an NVIDIA Tesla K80 accelerator, consisting of two GK210 chips with 2496 cores each (4992 cores in total) and 24 GiB of GDDR5 memory. Each M40 node is upgraded with an NVIDIA Tesla M40 accelerator, consisting of a GM200 chip with 3072 cores and 12 GiB of GDDR5 memory. The P100 nodes have two GPU cards installed each, providing 32 P100 GPUs in total. The P100 GPUs utilize GP100 chips providing 3584 cores, with 16 GiB GDDR5 RAM in each card. In order to make it easy for users to get started with the GPU nodes, we have developed a [CUDA appliance](#) that includes NVIDIA drivers as well as the CUDA framework.

| GPU | Chip | Cores per GPU | RAM per GPU | GPU per node | # of nodes |
|-------|------|---------------|-------------|--------------|------------|
| Tesla | GK | | 24 GiB | | |

| | | | | | |
|------------|--------|----------|--------------|---|----|
| K80 | 210 | 2496 × 2 | GDDR5 | 1 | 2 |
| Tesla M40 | GM 200 | 3072 | 12 GiB GDDR5 | 1 | 2 |
| Tesla P100 | GP100 | 3584 | 16 GiB GDDR5 | 2 | 16 |

The four FPGA nodes have a Nallatech 385A board with an Altera Arria 10 1150 GX FPGA (up to 1.5 TFlops), 8 GiB DDR3 on-card memory, and dual QSFP 10/40 GbE support. The Chameleon [FPGA User Guide](#) provides details for conducting experiments on this hardware.

The low-power systems are comprised of 8 low power Xeon servers (HP ProLiant m710p with one 4-core Intel Xeon E3-1284L v4 processor), 8 Atom servers (HP ProLiant m300 with one 8-core Intel Avoton-based System on a Chip), and 24 ARM servers (HP ProLiant m400 with one 8-core AppliedMicro X-gene System on a Chip). These are all delivered in a single HP Moonshot 1500 chassis.

For more information on how you can reserve these nodes, see the [heterogeneous hardware section](#) of the bare metal user’s guide.

13.5.3 Chameleon Cloud Charge Rates

It is important to fully understand the charge rates of your VM and storage use.

Chameleon has two types of limitations, introduced to promote fair resource usage to all:

Allocation:

Chameleon projects are limited to a per-project allocation currently set to 20,000 service units for 6 months. Allocations can be renewed or extended

Lease:

To ensure fairness to all users, resource reservations (leases) are limited to a duration of 7 days. However, an active lease within 48 hours of its end time

can be prolonged by up to 7 days from the moment of request if resources are available. To prolong a lease, click on the “Update Lease” button in the Reservations panel of the CHI OpenStack dashboard, and enter the additional duration requested in the “Prolong for” box including the unit suffix, e.g. “5d” for 5 days or “30m” for 30 minutes. If there is an advance reservation blocking your lease prolongation that could potentially be moved, you can interact through the users mailing list to coordinate with others users. Additionally, if you know from the start that your lease will require longer than a week and can justify it, you can [contact Chameleon staff via the ticketing system](#) to request a one-time exception to create a longer lease. The lease must be requested by the PI.

13.5.3.1 Service Units

Chameleon allocations can consist of several components of the system. Users can request allocation of individual compute nodes, storage servers, or complete Scalable Compute Units (SCUs) which contain compute servers, storage nodes, and an open flow switch.

Compute servers are allocated in Service Units (SUs), which equates to one hour of wall clock time on a single server (for virtual machines, an SU is 24 cores with up to 128GB of RAM). Note this unit differs from traditional HPC or cloud service units that are charged in core-hours; a Chameleon SU is a full server, as the type of experiments and performance measurements users may wish to do may be contaminated by sharing nodes.

Storage servers are also charged in SUs, at 2x the rate of compute servers (i.e., 1 hour allocation of 1 storage server == 2 SUs). SCUs are charged at the rate of 50 SUs per wall clock hour (42 compute servers, 4 storage nodes, plus one OpenFlow switch).

An allocation may make use of multiple SCUs, up to the size of the full testbed.

For example, a user wishing to provision a 10 node cluster +1 storage server for a 1 week experiment should budget $[(10 + 2) \text{ SUs per hour}] [7 \text{ days } 24 \text{ hours/day}] = 2,016 \text{ SUs}$ for that experiment.

SUs are charged the same regardless of use case. Hence, whether asking for bare

metal access, virtual machine access, or use of default images, the charge is the same — you are charged for the fraction of the resource your experiment occupies, regardless of the type of the experiment.

The basic principle for charging service units for Chameleon resources is to evaluate the amount of time a fraction of the resource is unavailable to other users. If a reservation is made through the portal for a particular date/time in the future, the user will be charged for this time regardless of whether the reservation is actually used, as the Chameleon scheduling system will have to drain the appropriate part of the system to satisfy the reservation, even if the nodes requested are not actually used. A reservation request may be cancelled in which case no charges will apply.

13.5.3.2 Project Allocation Size

Currently Chameleon is operating on a “soft allocation model” where each project, if approved, will receive a startup allocation of 20,000 SUs for six months that can be both recharged (i.e., more SUs can be added) and renewed (i.e., the duration can be extended) via submitting a renew/recharge request. This startup allocation value has been designed to respond to both PI needs (i.e., cover an amount of experimentation needed to obtain a significant result) and balance fairness to other users (it represents roughly 1% of testbed six months’ capacity). Requests for these startup projects will receive a fast track internal review (i.e., users can expect them to be approved within a few days).

A PI can apply for multiple projects/allocations; however, the number of held allocations will be taken into account during review.

As our understanding of user need grows we expect the Chameleon allocation model to evolve towards closer reflection of those needs in the form of more differentiated allocations that will allow us to give larger allocations to users for longer time.

Please be mindful to shutting down your VMS when not in use as even VMs that do not do any calculations get charged. In past classes we had students that did not shut down their VMs and within 2 weeks used up all SUs for the entire class of 70 students. We like to avoid this. In future cases we will assign the grade “F”

to such students, as is customary also in other universities.

13.5.4 Getting Started on Chameleon Cloud

We describe how you can get access to chameleon cloud under the assumption that you are a student or a researcher that joins an existing project on Chameleon cloud. You will need to follow the following steps:

13.5.4.1 Step 1: Create a Chameleon account

To get started using Chameleon you will need to [create a user account](#).

You will be asked to agree to the [Chameleon terms and conditions](#) which, among others, ask you to acknowledge the use of Chameleon in your publications.

Acknowledgement of support from the Chameleon project and the National Science Foundation should appear in any publication of material, whether copyrighted or not, that describes work which benefited from access to Chameleon cyberinfrastructure resources. The suggested acknowledgement is as follows: “Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation”.

As part of creating an account you may request PI status. However you are not a PI as you will be joining a project.

13.5.4.2 Step 2: Create or join a project

To use Chameleon, you will need to be associated with a [project](#) that is assigned an [allocation](#). This means that you either need to


1. [apply for a new project](#) or
2. [ask the PI of an existing Chameleon project to add you](#).

A project is headed by a project PI, typically [a faculty member or researcher scientist at a scientific institution](#). If you are a student we recommend that you ask your professor to work with you on creating a project. Please note that you must not create a project by yourself and that you indeed need to work with your

professor.

In case you need to do a project application typically consists of about one paragraph description of the intended research and takes one business day to process.

Enrolling you into an existing research or class project depends on the time availability of the project lead or professor of your class. It is important that you communicate your chameleon cloud account name to the project lead so they can easily add you. Make sure you really give them only your chameleon account name and potentially your organizational e-mail, Firstname, and Lastname so they can check you are eligible to get access.

 *Indiana University students that take the e516 and e616 classes will have to fill out a google form in which they communicate the chameleon cloud name. You can already apply for an account name, but do not apply for a project. If you nevertheless apply for a project, we will hear from the chameleon cloud administrators. As they do not like that because you have not followed the chameleon cloud project policies you will receive a grade deduction for the class.*

13.5.4.3 Step 3: Start using Chameleon

Now that you have enrolled and once you are added to the project by your project lead you can start using chameleon cloud. However be reminded that you ought to shut down the resources/VMs whenever they are not in use to avoid unnecessary charging. Remember the project has limited time on chameleon and any unused time will be charged against the project.

Chameleon provides two types of resources with links to their respective users guides next:

[Bare Metal User Guide](#) will tell you how to use Chameleon bare metal resources which provide strong isolation and allow you maximum control (reboot to new operating system, reboot the kernel, etc.)

[OpenStack KVM User Guide](#) will tell you how to get started with

Chameleon's OpenStack KVM cloud which is a multi-tenant environment providing weak performance isolation.

If you have any questions or encounter any problems, you can check out our [User FAQ](#), or [submit a ticket](#).

As part of the classes you will need to first pass a cloud *security* drivers licence test. The test is designed so that you think about gaining access to a VM securely and how to properly secure the VM. Once passed, access is typically provided by midterm time. You are not allowed to constantly run VM's and must shut them down if not in use. You will get point deductions if we detect you do not obey by this rule. We have access to log files about your VM usage.

13.5.5 OpenStack Virtual Machines

OpenStack is an Infrastructure as a Service (IaaS) platform that allows you to create and manage virtual environments. Chameleon provides an installation of OpenStack version 2015.1 (Kilo) using the KVM virtualization technology.

Since the KVM hypervisor is used on this cloud, any virtual machines you upload must be compatible with KVM.

This section provides basic information about how to use the OpenStack web interface and provides some information specific to using OpenStack KVM on Chameleon.

13.5.5.1 Web Interface

An easy way to use OpenStack KVM on Chameleon is via the [OpenStack web interface](#) also known as Horizon. You log into the web interface using your Chameleon username and password. If you change your Chameleon password in the portal, that change will propagate to the OpenStack KVM interface in about 5 minutes. See [Figure 69](#)

The initial log in page appears as:



Figure 69: Chameleon login

After a successful log in, you will see the Overview page as shown next. This page provides a summary of your current and recent usage and provides links to various other pages. Most of the tasks you will perform are done via the menu on the lower left and will be described next. One thing to note is that on the left, your current project is displayed. If you have multiple Chameleon projects, you can change which of them is your current project. All of the information displayed and actions that you take apply to your current project. So in the screen shot [Figure 70](#), the quota and usage apply to the current project you have selected and no information about your other projects is shown. See [Figure 70](#)

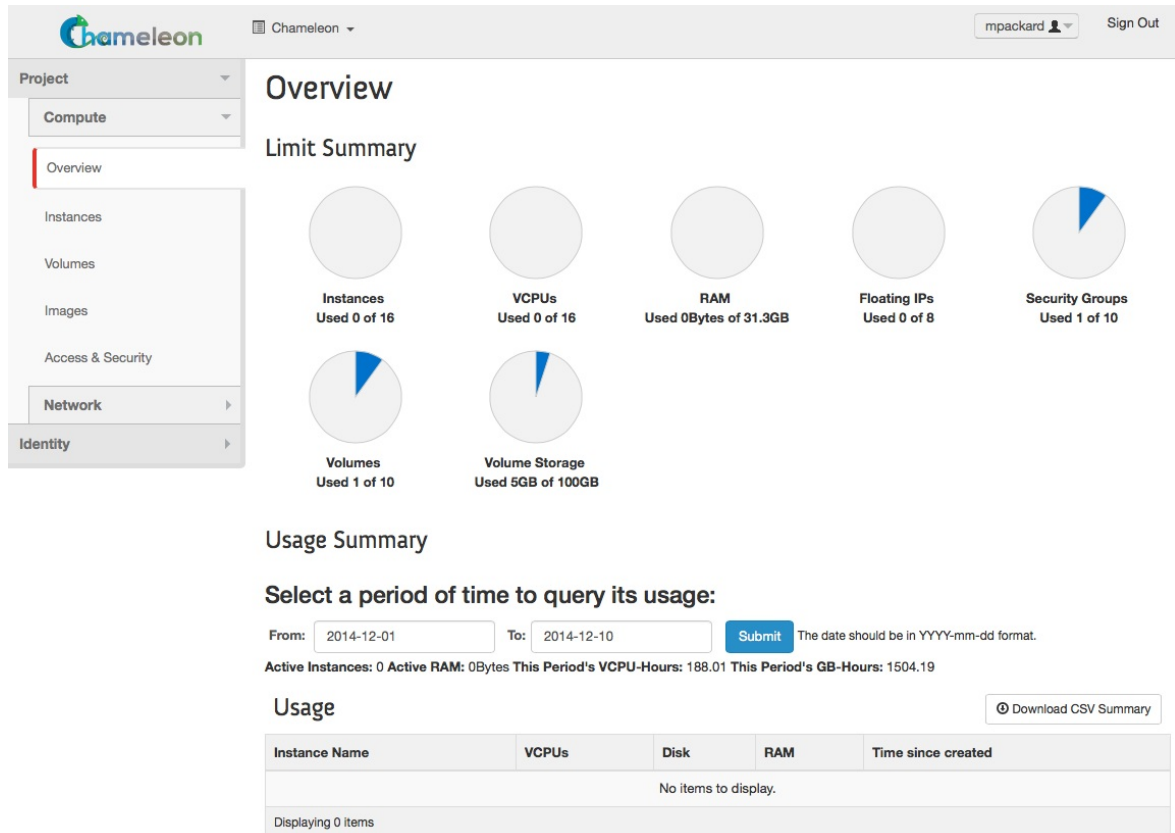


Figure 70: Overview page

13.5.5.1.1 Managing Virtual Machine Instances

One of the main activities you'll be performing in this web interface is the management of virtual machines, or instances. You do this via the Instances page that is reachable from the menu in the lower left of the Overview page. An example Instances page is shown next. For instances that you have running, you can click on the name of the instance to get more information about it and to access the VNC interface to the console. The dropdown menu to the left of the instance lets you perform a variety of tasks such as suspending, terminating, or rebooting the instance. See [Figure 71](#)

The screenshot shows the Chameleon web interface. On the left is a navigation sidebar with categories like Project, Compute, Overview, Instances (highlighted), Volumes, Images, Access & Security, Network, and Identity. The main content area is titled 'Instances' and contains a table of running virtual machines. Above the table are controls for filtering and actions like 'Launch Instance', 'Soft Reboot Instances', and 'Terminate Instances'.

| Instance Name | Image Name | IP Address | Size | Key Pair | Status | Availability Zone | Task | Power State | Time since created | Actions |
|------------------------------------------|-------------------------|-------------------------------|----------|----------|--------|-------------------|------|-------------|--------------------|-----------------|
| <input type="checkbox"/> Ubuntu instance | Ubuntu-Server-14.04-LTS | 192.168.0.53
129.114.32.26 | m1.small | mpackard | Active | Alamo | None | Running | 0 minutes | Create Snapshot |
| <input type="checkbox"/> CentOS image | CentOS-7 | 192.168.0.52
129.114.32.27 | m1.small | mpackard | Active | Alamo | None | Running | 0 minutes | Create Snapshot |

Displaying 2 Items

Figure 71: Virtual Machine instances

The Instances page also lets you create new virtual machines by using the ‘Launch Instance’ button in the upper-right. When you click this button, a dialog window pops up. In the first ‘Details’ tab, you select the ‘Instance Boot Source’ of the instance, which is either an ‘Image’, a ‘Snapshot’ (an image created from a running virtual machine), or a ‘Volume’ (a persistent virtual disk that can be attached to a virtual machine). If you select ‘Boot from image’, the Image Name dropdown presents a list of virtual machine images that we have provided, that other Chameleon users have uploaded and made public, or images that you have uploaded for yourself. If you select ‘Boot from snapshot’, the Instance Snapshot dropdown presents a list of virtual machine images that you have created from your running virtual machines.

On the Details tab, you also provide a name for this instance (to help you identify instances that you are running), and select the amount of resources (Flavor) to allocate to the instance. If you select different flavors from the Flavor dropdown, their characteristics are displayed on the right. See [Figure 72](#)

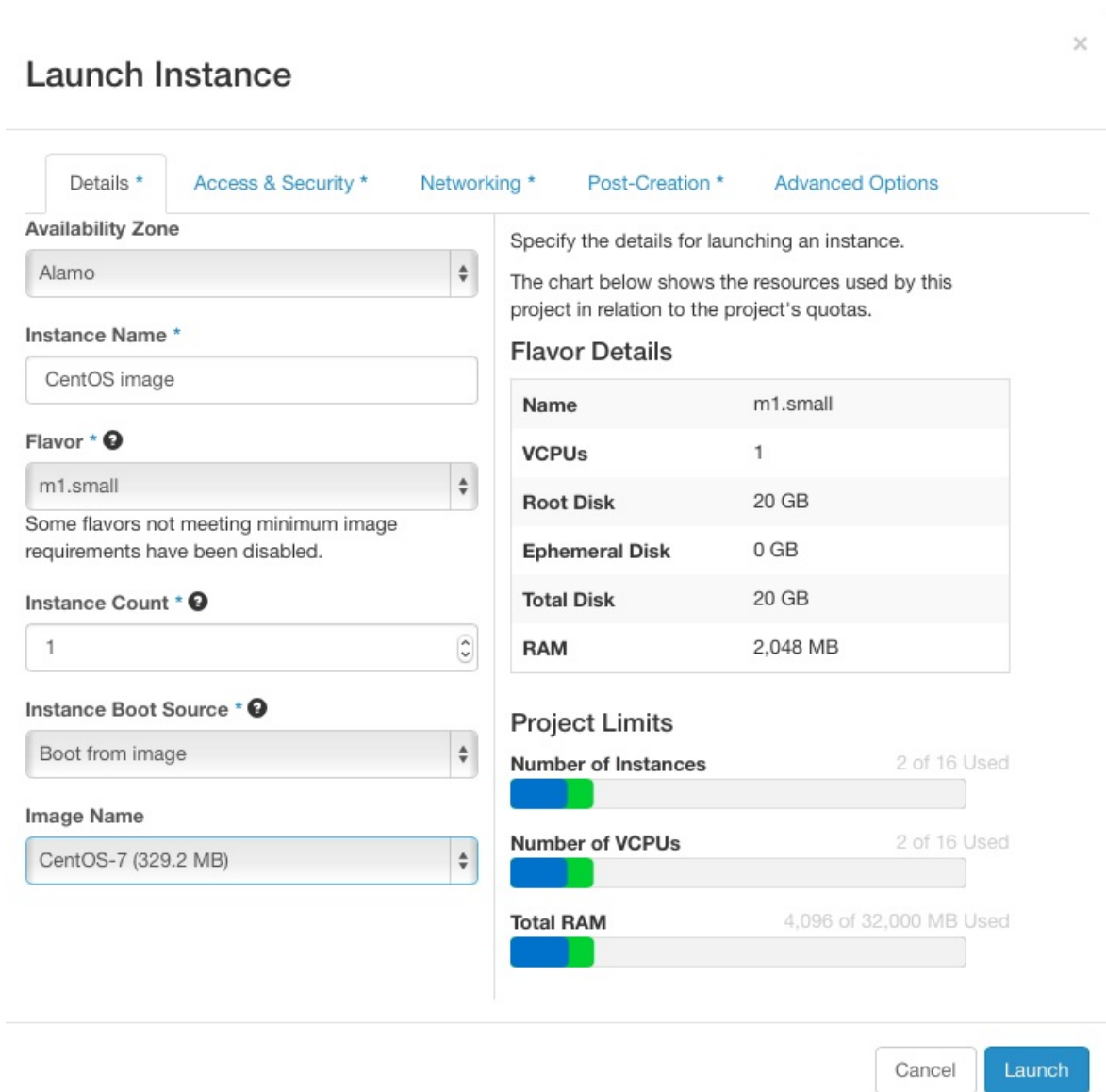


Figure 72: Launcher window

The next tab is 'Access & Security', where you select an SSH keypair that will be inserted into your virtual machine. These keypairs can be uploaded via the main 'Access & Security' section. You will need to select a keypair here to be able to access an instance created from one of the public images Chameleon provides. These images are not configured with a default root password and you will not be able to log in to them without configuring an SSH key. See [Figure 73](#)

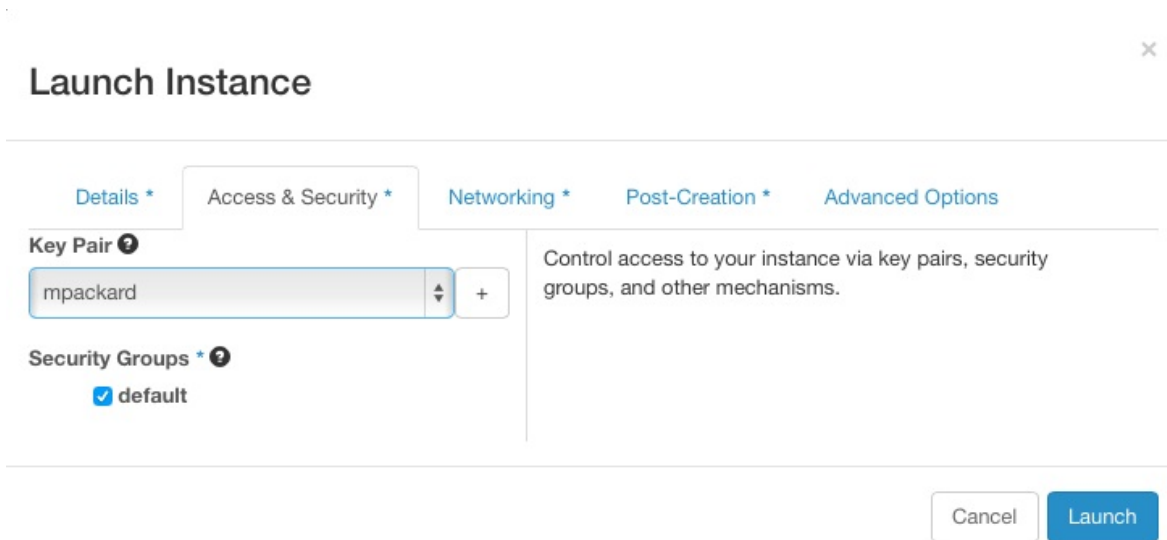


Figure 73: Access window

Next is 'Networking', where you select which network should be associated with the instance. Click the + next to your project's private network (PROJECT_NAME-net), not ext-net. See [Figure 74](#)

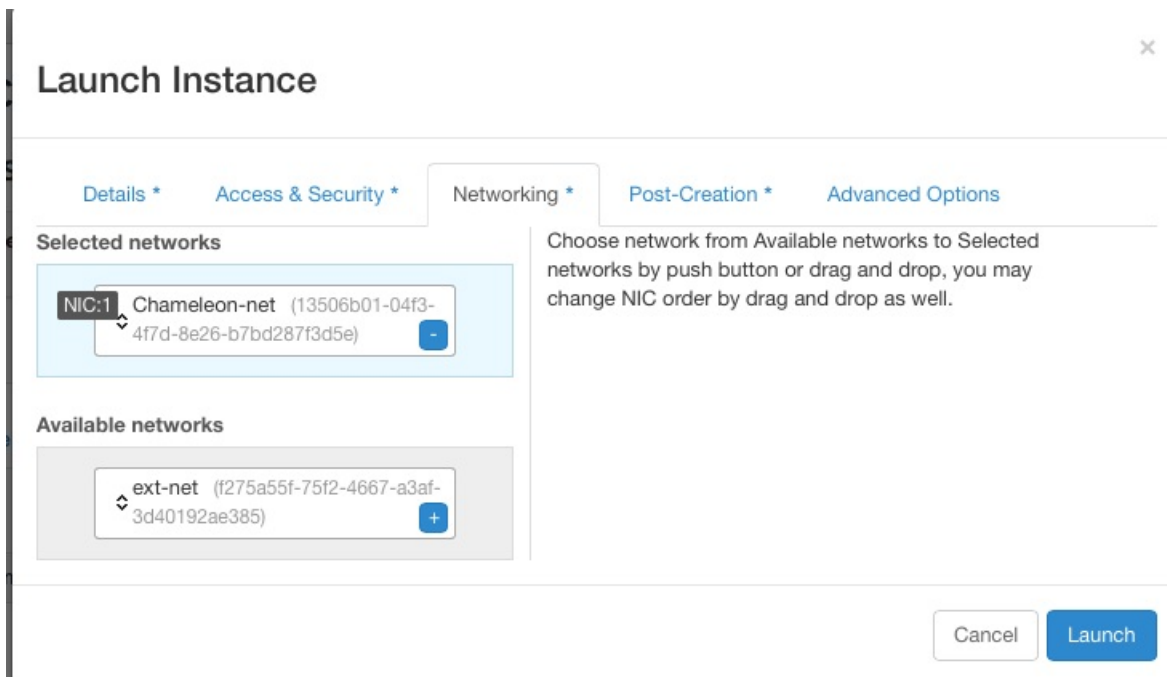


Figure 74: Networking window

Once you do this, you can Launch your instance and the Instances page will show progress as it starts.

If you would like to assign a public IP address to your VM, you can do that while it is booting up. Click on the dropdown under *Actions* and choose *Associate Floating IP*. Choose an IP from the *IP Address* menu and click *Associate*. If there are no addresses available, click the + and follow the prompts to add one. See [Figure 75](#)

Manage Floating IP Associations

IP Address *

IP Address *

129.114.32.32

Port to be associated *

test: 192.168.0.57

Select the IP address you wish to associate with the selected instance.

Cancel Associate

Figure 75: Floating IP window

OpenStack injects your SSH key into the VM and you can use the corresponding private SSH key to log in to the VM. You will need to use the public IP assigned to your VM to connect from outside of Chameleon, or connect through an existing instance that both a public and private IP.

Note that the images we provide do not allow SSH into the root account. For root access, SSH into the instance as user 'cc' and then use the *sudo* command to become root.

We have enabled auto-login for the cc user on the console of our supported images. This should aid in debugging if you are unable to reach the instance via ssh for some reason. See [Figure 76](#)



Figure 76: Console

13.5.5.1.2 Snapshots

The instance list page shown previously has an option ‘Create Snapshot’ that allows you to save a copy of the disk contents of a running virtual machine. This allows you to start new virtual machines in the future that are identical to this one and is an easy way to save any changes you make to a running virtual machine.

13.5.5.1.3 Firewall (Access Security)

Each project has control over their own firewall settings for their instances. At minimum you’ll probably want to allow SSH access so you can reach your instances.

To enable this traffic, you need to configure the security group used by your virtual machine. You can see a list of your security groups using the “Access & Security” link on the left. See [Figure 77](#)

Access & Security

Security Groups Key Pairs Floating IPs API Access

Security Groups

[+ Create Security Group](#) [X Delete Security Groups](#)

| <input type="checkbox"/> | Name | Description | Actions |
|--------------------------|---------|-------------|------------------------------|
| <input type="checkbox"/> | default | default | Manage Rules |

Displaying 1 item

Figure 77: Security groups

To edit a security group, click on “Edit Rules”. This opens a page showing the existing rules in the security group. See [Figure 78](#)

Manage Security Group Rules: default

Security Group Rules

[+ Add Rule](#) [X Delete Rules](#)

| <input type="checkbox"/> | Direction | Ether Type | IP Protocol | Port Range | Remote | Actions |
|--------------------------|-----------|------------|-------------|------------|------------------|-----------------------------|
| <input type="checkbox"/> | Ingress | IPv4 | Any | - | default | Delete Rule |
| <input type="checkbox"/> | Egress | IPv6 | Any | - | ::/0 (CIDR) | Delete Rule |
| <input type="checkbox"/> | Egress | IPv4 | Any | - | 0.0.0.0/0 (CIDR) | Delete Rule |
| <input type="checkbox"/> | Ingress | IPv6 | Any | - | default | Delete Rule |
| <input type="checkbox"/> | Ingress | IPv4 | ICMP | - | 0.0.0.0/0 (CIDR) | Delete Rule |
| <input type="checkbox"/> | Ingress | IPv4 | TCP | 22 (SSH) | 0.0.0.0/0 (CIDR) | Delete Rule |

Displaying 6 items

Figure 78: Editing a security group

Click on “Add Rule” and choose the *SSH* rule from the list, and click *Add*. Modifications are automatically propagated to the OpenStack cloud. Feel free to add other rules as necessary. See [Figure 79](#)

x

Add Rule

Rule *

SSH

Remote * ?

CIDR

CIDR ?

0.0.0.0/0

Description:

Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:

Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.

Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.

Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.

Cancel

Add

Figure 79: Add a security group

13.5.5.2 OpenStack REST Interfaces

The OpenStack REST Interfaces are supported on Chameleon over secure HTTP connections. You can download your OpenStack credentials file from the web interface via the “Access & Security” link in the left of any page and then click on the “API Access” link on the top.

You can then install the OpenStack command line clients following [these instructions](#). If using pip, we recommend setting up a virtualenv.

The SSL certificate used by Chameleon is trusted by most operating systems, so you should not have to provide any extra options to OpenStack commands, i.e. “nova list” should work. If your command-line tool complains about the certificate, [download the Mozilla CA bundle from the cURL website](#) and run the

OpenStack client tools with the `--os-cacert cacert.pem` arguments.

13.5.5.3 Downloading and uploading data

You can use the OpenStack command line clients to download data from and upload data to Chameleon clouds. Configure your environment by following the “OpenStack REST Interfaces” previous section, then use the following commands:

- `glance image-download` to download images and snapshots from Glance
- `glance image-create` to upload images and snapshots to Glance
- `cinder upload-to-image` to convert a Cinder volume to a Glance image
- `cinder create [--image-id <image-id>] [--image <image>]` to create a Cinder volume from a Glance image

13.5.6 Cloudfish OpenStack Command Line Interface

OpenStack on Chameleon delivers KVM based compute resources to provision virtual machines. It provides various image types on which we can deploy tools and software needed for the class and projects. We will you through the basic steps of getting access to OpenStack Chameleon cloud under the class allocation. Next, we will introduce you the cloudfish command line tools which you can use in your projects. Naturally using the GUI for your projects is not sufficient as setting up your environment will need steps to be executed by hand which is not sufficient. It is a goal of this class that you create your environment in a reproducible fashion via scripts. Hence, although the Web interface called OpenStack Horizon is initially attractive, we should make sure to move on to the commandline interfaces. Furthermore, it is often difficult to resolve technical issues as the command line tools generate full debugging messages in case of issues and copy and past into help windows is much easier and efficient than copy and past incomplete screenshots.

One important factor for using cloudfish shell is that it not only works for chameleon cloud but also for AWS and Azure. We are hoping to add Google also which is already in our preliminary code.

-  [Cloudfish Commandline Interface Demonstration](#)

13.5.6.1 Instalation of Cloudmesh Client

We discuss how to install cloudmesh in the [Cloudmesh manual] [https://cloudmesh.github.io/cloudmesh-manual/installation/install.html]

We assume that your public key is located at `~/.ssh/id_rsa.pub`

We assume you have the file `~/.cloudmesh/cloudmesh.yaml` that is created during the instalation process. Please also make sure that the file `~/.cloudmesh/names.yaml` Is properly configured for the class. Typically it will look like

```
path: /Users/grey/.cloudmesh/name.yaml
schema: NNN-accountname-{counter}
counter: 1
```

Where NNN is the last three gigits from your hid that we place in github and for accountname, please chose your chameleon account name. If you are not taking any of our classes and you do not have a github directory that we created for you, please use

```
path: /Users/grey/.cloudmesh/name.yaml
schema: accountname-{counter}
counter: 1
```

instead. Whenever you start a new vm, the counter of the vm gets increased, guranteeing a unique virtual machine name across all colaborators and your own virtual machines.

We also assume you have called the command

```
cms init
```

and are running the MongoDB cloudmesh service which you can check with

```
$ cms admin mongo status
```

Once you install cloudmesh you need to modify the `~/.cloudmesh/cloudmesh.yaml` file to add your username and password. Make sure to properly protect this file as discussed in the manual.

To add the username and password, you can use an editor, or execute on the commandline with the commands

```
$ cms config set chameleon.OS_USERNAME=YOURUSERNAME
```

```
$ cms config set chameleon.OS_PASSWORD=YOURPASSWORD
```

They will change the values in the yaml file at

- `cloudmesh.cloud.chameleon.credentials.`

Next test out if you can see some images with

```
cms image list --refresh
```

You will see a table similar to

```
+-----+-----+-----+-----+-----+-----+
| Name           | Size (Bytes) | MinDisk (GB) | MinRam (MB) | Status | Driver |
+-----+-----+-----+-----+-----+-----+
CC-Ubuntu18.04	982843392	0	0	ACTIVE	openstack
CC-Ubuntu16.04	844759040	0	0	ACTIVE	openstack
CC-Ubuntu18.04-20190822	982056960	0	0	ACTIVE	openstack
CC-Ubuntu16.04-20190822	844824576	0	0	ACTIVE	openstack
...					
```

To see the flavors or sizes, you can use

```
cms flavor list flavor --refresh
```

Which will return something like

```
+-----+-----+-----+-----+
| Name           | VCPUS | RAM | Disk |
+-----+-----+-----+-----+
m1.tiny	1	512	1
m1.small	1	2048	20
m1.medium	2	4096	40
m1.large	4	8192	80
m1.xlarge	8	16384	160
storage.medium	1	4096	2048
m1.xxlarge	8	32768	160
m1.xxxlarge	16	32768	160
+-----+-----+-----+-----+
```

Cloudmesh reads the preset variables in the `cloudmesh.yaml` file to start new virtual machines. To see them you can look at the yaml file or use the command

```
$ cms config get chameleon.default
```

To start a VM simply use

```
cms vm boot
```

You will see something similar to

```
# -----
# Create Server
# -----
Name: benchmark-gregor-vm-684
```

```
User: cc
IP: 129.114.33.243
Image: CC-Ubuntu14.04
Size: m1.small
Public: True
Key: gregor
location: None
timeout: 360
secgroup: default
group: cloudmesh
groups: ['cloudmesh']
```

To log into the vm you can use

```
cms ssh
```

To set a different vm, you could use the command line parameters that you can find out with

```
cms vm help
```

but in case you always want to use the same parameters it is much more convenient to use our `config set` command with

```
$ cms config set cloud.chameleon.default.size=CC-Ubuntu18.04
$ cms config set cloud.chameleon.default.image=m1.small
$ cms config set cloud.chameleon.default.username=cc
```

On chameleon cloud images with CC are chameleon cloud sanctioned images. They include some monitoring extensions and use the username `cc` for login.

13.5.6.2 Floating IP Address

We have configured cloudmesh to automatically assign a floating ip address so you can use that to log into the vm.

to view it, you can use the command

```
$ cms vm list --refresh
```

To delete the vm simply say

```
$ cms vm delete
```

13.5.7 OpenStack Horizon Graphical User Interface

-  [Horizon, Starting a VM](#)

13.5.7.1 Configure resources

Once your lease is started, you are almost ready to start an instance. But first, you need to make sure that you will be able to connect to it by setting up a key pair. This only has to be done once per user per project.

Go to Project > Compute > Access & Security, then select the Key Pairs tab. See [Figure 80](#)

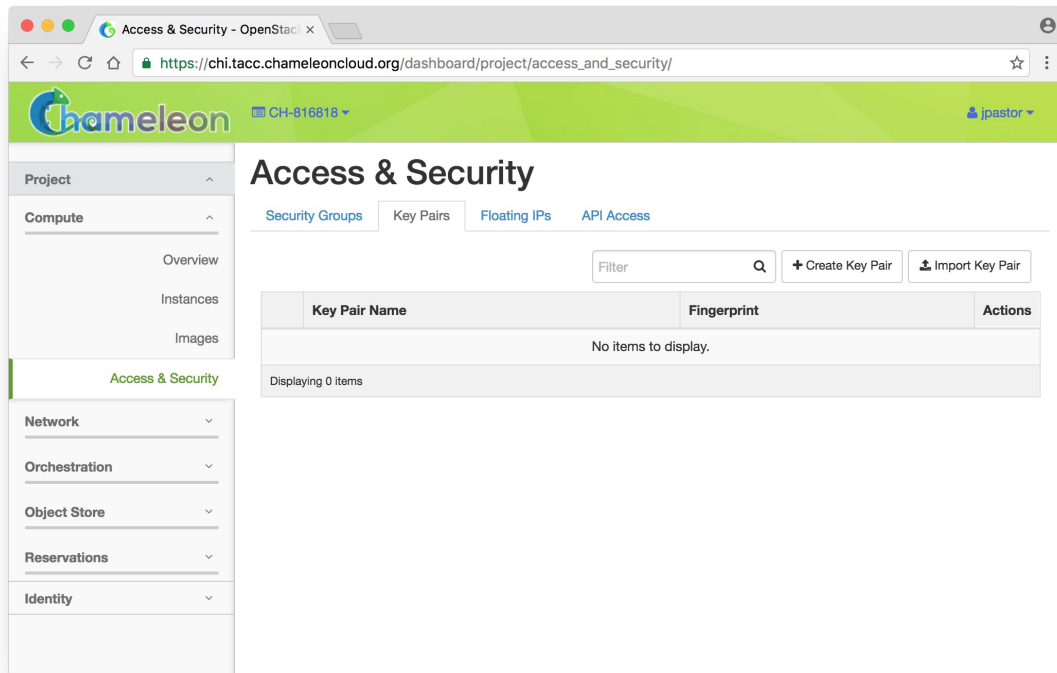


Figure 80: Key Pairs Tab

Here you can either ask OpenStack to create an SSH key pair for you (via the “Create Key” Pair button), or, if you already have an SSH key pair on your machine and are happy to use it, click on “Import Key Pair”.

If you chose to import a key pair, you will be asked to enter a name for the key pair, for example laptop. In the “Public Key” box, copy the content of your SSH public key. Typically it will be at `~/.ssh/id_rsa.pub`. On Mac OS X, you can run in a terminal: `cat ~/.ssh/id_rsa.pub | pbcopy` It copies the content of the public key to your copy/paste buffer. Then you can simply paste in the “Public Key” box. See [Figure 81](#)

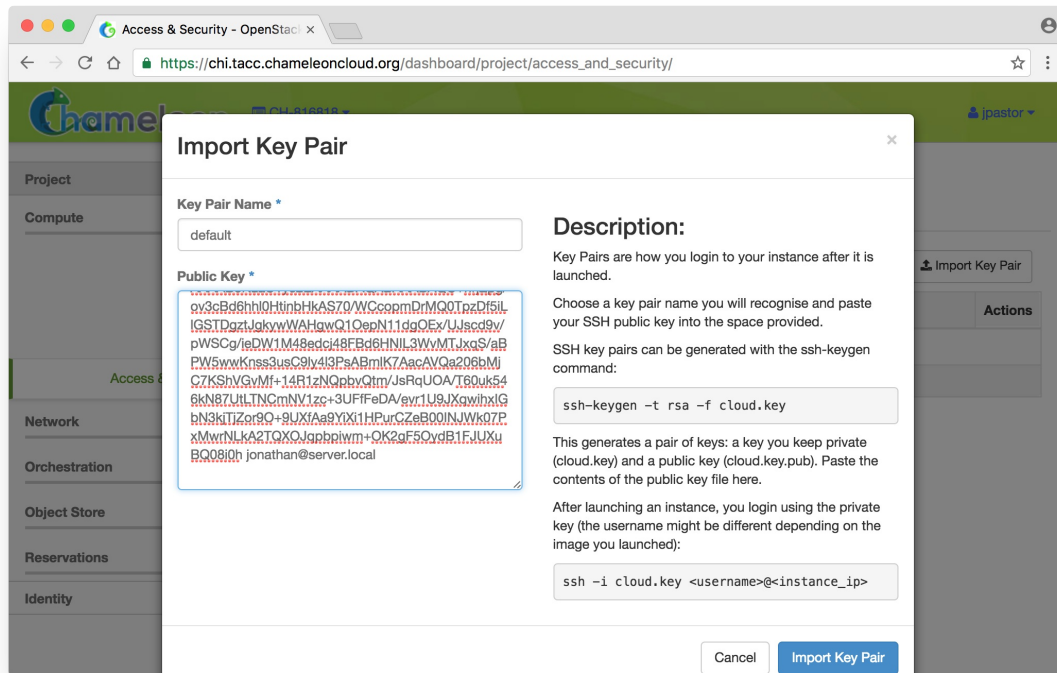


Figure 81: Public Key

Then, click on the blue “Import Key Pair” button. This should show you the list of key pairs, with the one you just added. See [Figure 82](#)

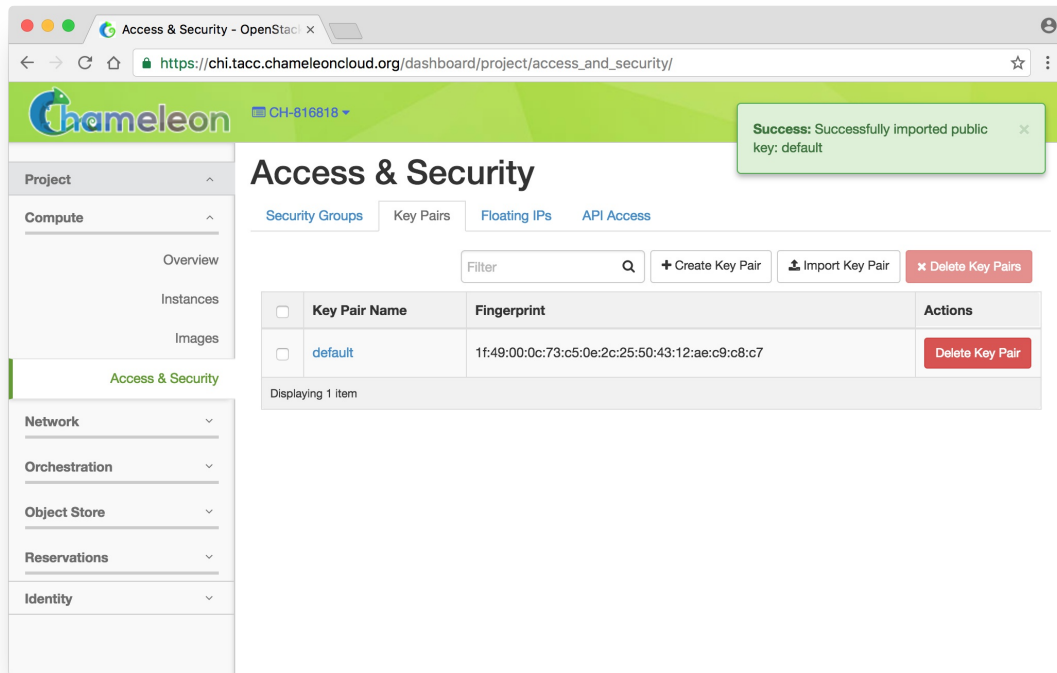


Figure 82: Import key pair

For those already familiar with OpenStack, note that Security Groups are not functional on bare-metal. All instances ports are open to the Internet and any security group rule you add will not be respected.

Now, go to the “Instances” panel. See [Figure 83](#)

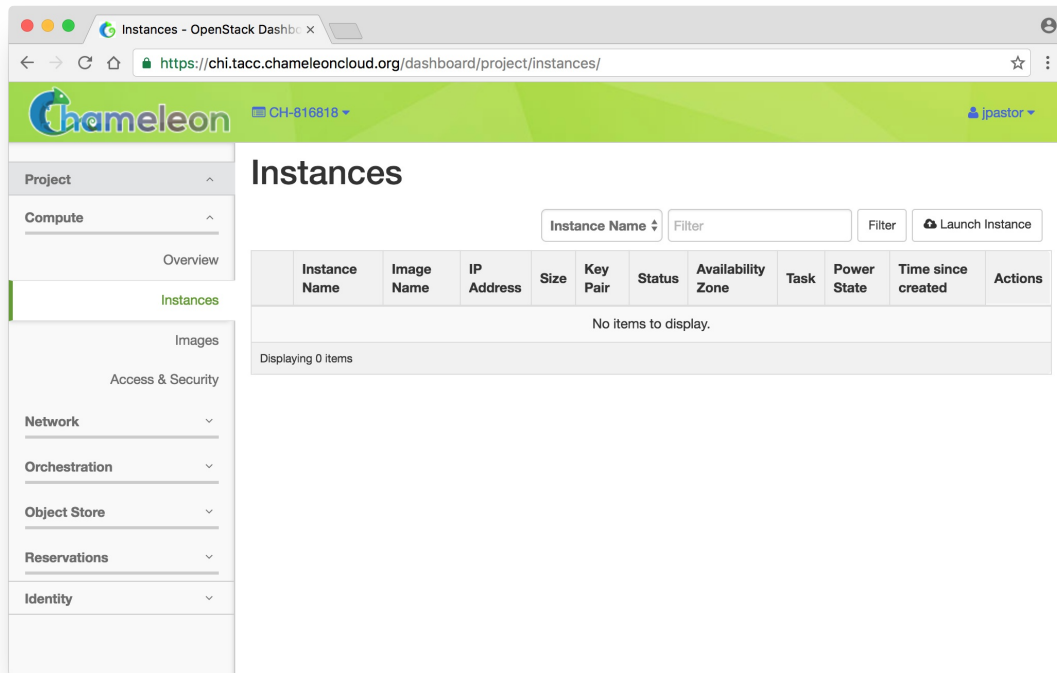


Figure 83: VM Instances

Click on the “Launch Instance” button in the top right corner. Select a reservation in the Reservation box, pick an instance name (in this example my-first-instance) and in the Image Name list select our default environment named CC-CentOS7. If you have multiple key pairs registered, you need to select one in the “Access & Security” tab. Finally, click on the blue “Launch” button. See [Figure 84](#)

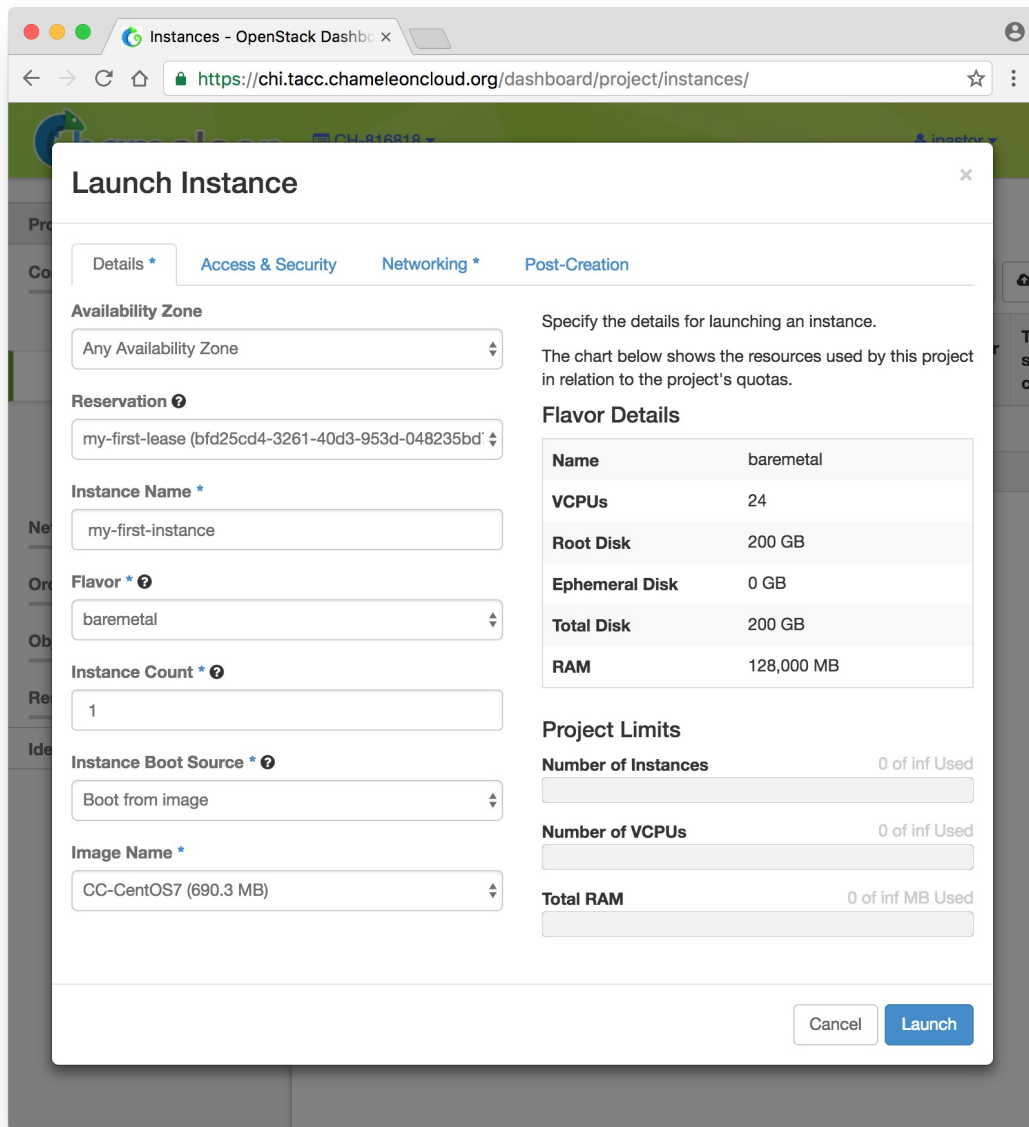


Figure 84: Launch a VM

The instance will show up in the instance list, at first in Build status. It takes a few minutes to deploy the instance on bare-metal hardware and reboot the machine. See [Figure 85](#)

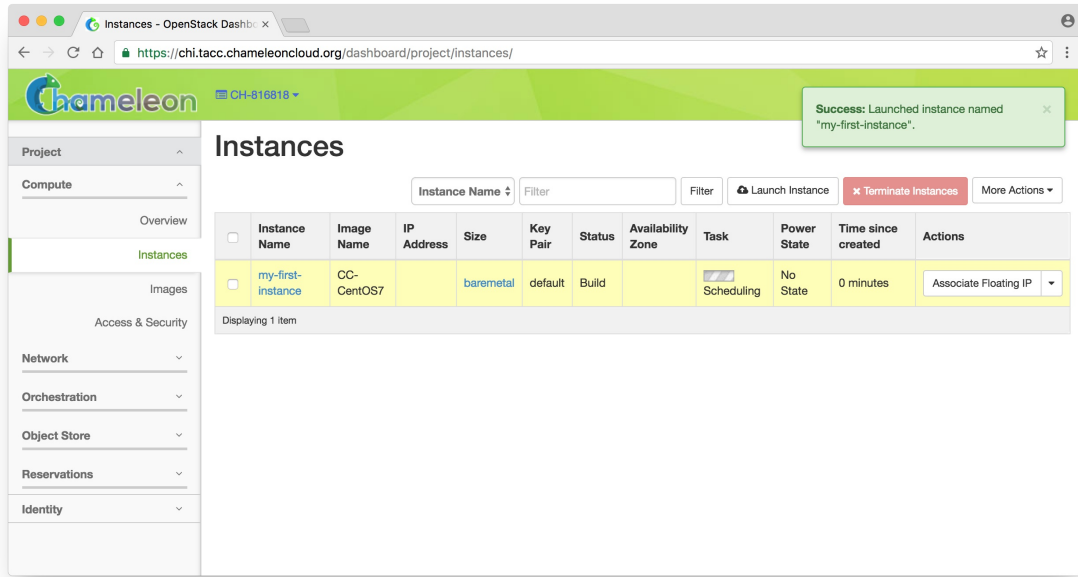


Figure 85: Status Window (a)

After a few minutes the instance should become in Active status and the Power State should be Running. See [Figure 86](#)

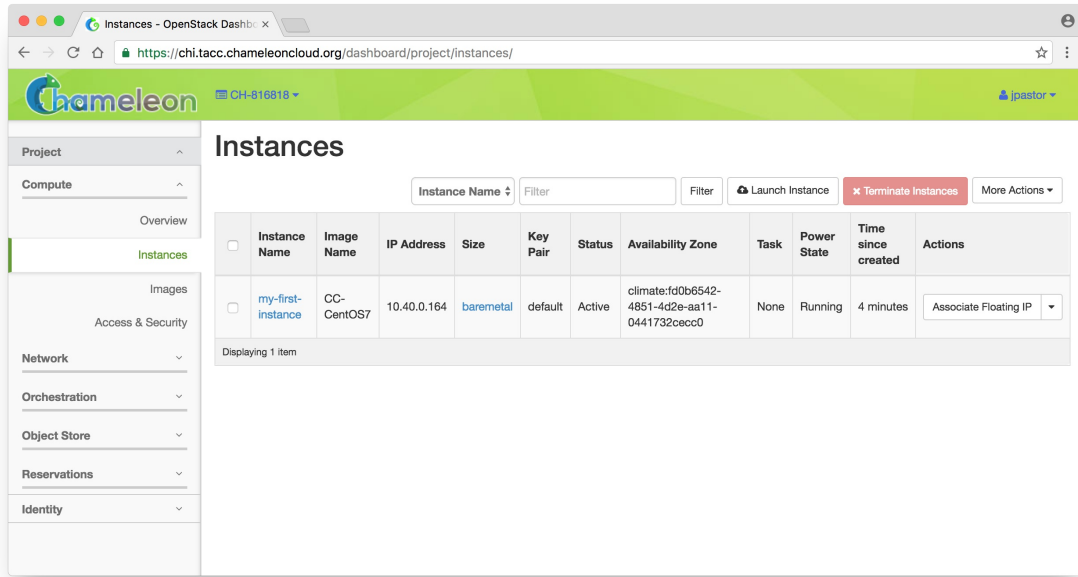


Figure 86: Status Window (b)

At this point the instance might still be booting: it might take a minute or two to actually be accessible on the network and accept SSH connections. In the meantime, you can attach a floating IP to the instance. Click on the “Associate Floating IP” button. You should get a screen like this one: [Figure 87](#)

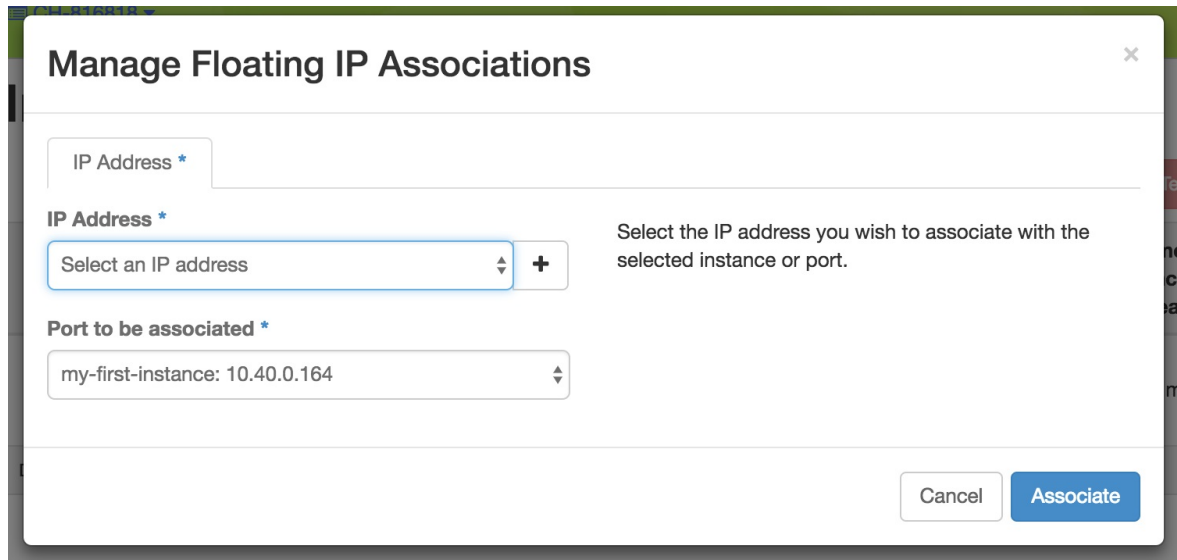


Figure 87: Floating IP

If there are no unused floating IP already allocated to your project, click on the + button. In the window that opens, select the ext-net pool if not already selected

by default and click on the blue Allocate IP button. See [Figure 88](#)



Figure 88: Allocate the IP

You will be returned to the previous window. The correct value for “Port to be associated” should already be selected, so you only have to click on “Associate”. See [Figure 89](#)

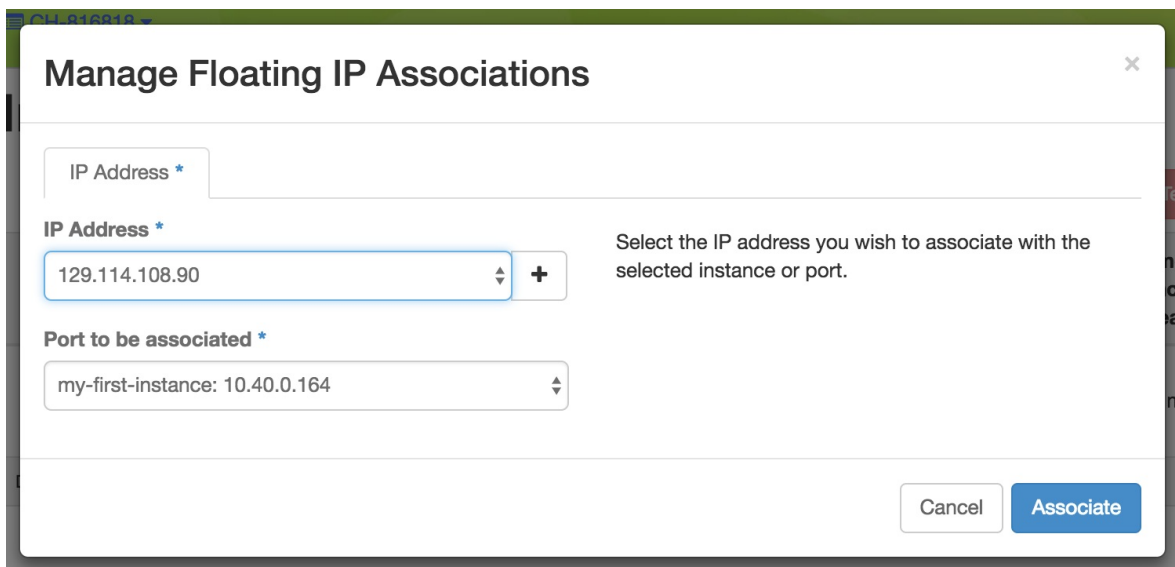


Figure 89: Associate the IP

This should send you back to the instance list, where you can see the floating IP attached to the instance (you may need to refresh your browser to see the floating IP). See [Figure 90](#)

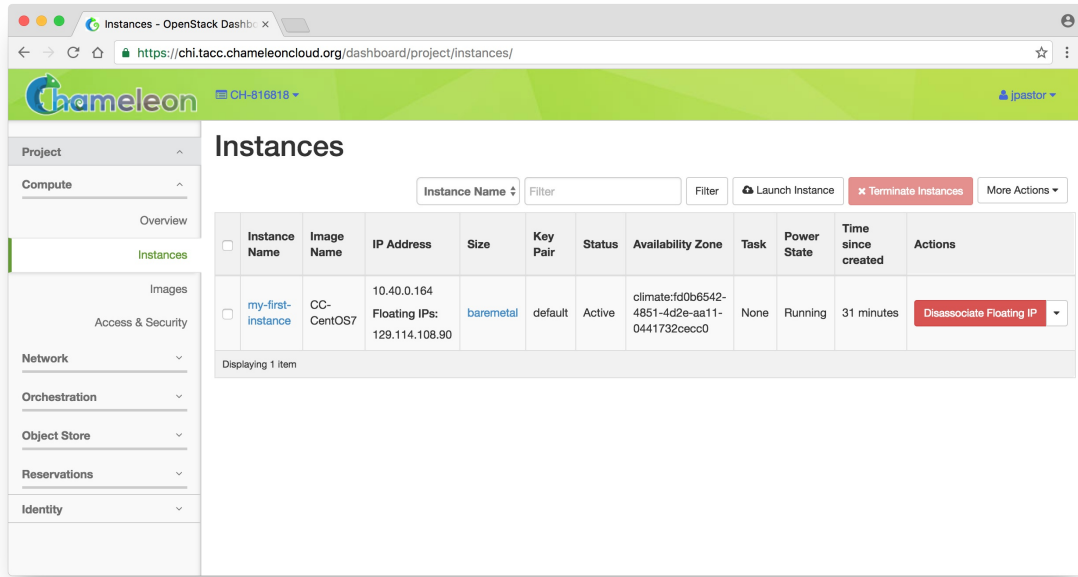


Figure 90: Status of the IP Association

13.5.7.2 Interact with resources

Now you should be able to connect to the instance via SSH using the cc account. In a terminal, type `ssh cc@floating_ip`, in our example this would be

```
$ ssh cc@130.202.88.241
```

SSH will probably tell you:

```
The authenticity of host }130.202.88.241
(130.202.88.241) cannot be established. RSA key fingerprint
is 5b:ca:f0:63:6f:22:c6:96:9f:c0:4a:d8:5e:dd:fd:eb.
Are you sure you want to continue connecting (yes/no)?
```

Type yes and press Enter. You should arrive to a prompt like this one:

```
[cc@my-first-instance ~]$
```

If you notice SSH errors such as connection refused, password requests, or failures to accept your key, it is likely that the physical node is still going through the boot process. In that case, please wait before retrying. Also make sure that you use the **cc** account. If after 10 minutes you still cannot connect to the machine, please [open a ticket with our help desk](#).

You can now check whether the resource matches its known description in the

resource registry. For this, simply run: `sudo cc-checks -v`

As of 03/30/2018, the `cc-checks` command may not work on the images in Chameleon cloud. You may have to ignore (not run) this command. See [Figure 91](#)

```
priteau — cc@test-new-image:~ — ssh — 55x56
cc@test-new-image:~
Chassis
  OK should have the correct serial number
  OK should have the correct manufacturer
  OK should have the correct product name

Disk
  OK should have the correct name
  OK should have the correct size
  OK should have the correct model
  OK should have the correct revision
  OK should have the correct vendor

Virtual Hardware
  OK should have the good driver

Memory
  OK should have the correct size

Network
  OK should be the correct interface name
  OK should have the correct Driver
  OK should have the correct Mac Address
  OK should have the correct Rate
  OK should have the correct version
  OK should have the correct mounted mode
  OK should not be a management card
  OK should be the correct interface name
  OK should have the correct Driver
  OK should have the correct Mac Address
  OK should have the correct Rate
  OK should have the correct version
  OK should have the correct mounted mode
  OK should not be a management card

OS
  OK should be the correct name
  OK should be the correct kernel version
  OK should be the correct version

Processor
  OK should have the correct frequency
  OK should be of the correct instruction set
  OK should be of the correct model
  OK should be of the correct version
  OK should have the correct vendor
  OK should have the correct description
  OK should have the correct L1i
  OK should have the correct L1d
  OK should have the correct L2
  OK should have the correct L3

Rights on /tmp
  OK should have mode 41777
[cc@test-new-image ~]$ echo $?
0
[cc@test-new-image ~]$ █
```

Figure 91: cc-check program

The `cc-checks` program prints the result of each check in green if it is successful and red if it failed.

You can now run your experiment directly on the machine via SSH. You can run commands with root privileges by prefixing them with `sudo`. To completely switch user and become root, use the `sudo su - root` command.

13.5.7.2.1 Snapshot an instance

All instances in Chameleon, whether KVM or bare-metal, are running off disk images. The content of these disk images can be snapshotted at any point in time, which allows you to save your work and launch new instances from updated images later.

While OpenStack KVM has built-in support for snapshotting in the Horizon web interface and via the command line, bare-metal instances require a more complex process. To make this process easier, we developed the [cc-snapshot](#) tool, which implements snapshotting a bare-metal instance from command line and uploads it to Glance, so that it can be immediately used to boot a new bare-metal instance. The snapshot images created with this tool are whole disk images.

For ease of use, *cc-snapshot* has been installed in all the appliances supported by the Chameleon project. If you would like to use it in a different setting, it can be downloaded and installed from the [github repository](#).

Once `cc-snapshot` is installed, to make a snapshot of a bare-metal instance, run the following command from inside the instance:

```
sudo cc-snapshot <snapshot_name>
```

You can verify that it has been uploaded to Glance by running the following command:

```
glance image-list
```

If you prefer to use a series of standard Unix commands, or are generally interested in more detail about image management, please refer to our [image](#)

[management guide](#).

13.5.7.3 Use FPGAs

Consult the [dedicated page](#) if you would like to use the FPGAs available on Chameleon.

13.5.7.4 Next Step

Now that you have created some resources, it is time to interact with them! You will find instructions to the next step by visiting the following link:

- [Monitor resources and collect results](#)

13.5.8 OpenStack HEAT

Deploying an MPI cluster, an OpenStack installation, or any other type of cluster in which nodes can take on multiple roles can be complex: you have to provision potentially hundreds of nodes, configure them to take on various roles, and make them share information that is generated or assigned only at deployment time, such as hostnames, IP addresses, or security keys. When you want to run a different experiment later you have to redo all this work. When you want to reproduce the experiment, or allow somebody else to reproduce it, you have to take very precise notes and pay great attention to their execution.

To help solve this problem and facilitate reproducibility and sharing, the Chameleon team configured a tool that allows you to deploy complex clusters with “one click”. This tool requires not just a simple image (i.e., appliance) but also a document, called a template, that contains the information needed to orchestrate the deployment and configuration of such clusters. We call this image + template combination complex appliance because it consists of more than just the image (i.e., appliance).

13.5.8.1 Supporting Complex Appliances

In a nutshell, complex appliances allow you to specify not only what image you want to deploy but also on how many nodes you want to deploy that image, what

roles the deployed instances should boot into (such as e.g., head node and worker node in a cluster), what information from a specific instance should be passed to another instance in that complex appliance, and what scripts should be executed on boot so that this information is properly used for configuring the “one click” cluster. For example, a Network File System (NFS) appliance that we will use as an example in this guide, might specify deployment on three nodes, out of which one will be configured as head node and others as worker nodes, the information passed between the images will be hostname of the head node, and the scripts executed on the worker nodes on boot will put that hostname in the fstab file. As you can tell from this description, images used for complex appliances are typically configured such that they can be booted into any role required on the one-click cluster we are booting; in this case the image will have both the software for NFS server node and client node.

Since complex appliances in Chameleon are currently implemented using the [OpenStack Heat](#) orchestration service, we will be using OpenStack terminology and features to work with them. The templates described previously are YAML files using the [Heat Orchestration Template \(HOT\) format](#) (Heat also supports the AWS CloudFormation template format, but this is not covered here). A deployed complex appliance is referred to as a “stack” – just as a deployed single appliance is typically referred to as an “instance”. This guide will tell you all you need to know in order to use and configure complex appliances on Chameleon; if you would like to know more about Heat, please refer to its [official documentation](#).

13.5.8.2 Chameleon Appliance Catalog

Our [Appliance Catalog](#) has several complex appliances for popular technologies that people want to deploy such as OpenStack or MPI or even more advanced deployments such as efficient SR-IOV enabled MPI in KVM virtual machines. We also provide common building blocks for cluster architectures, such as an NFS share. Complex appliances are identified by a badge in their top-right corner representing a group of machines, as shown in [Figure 92](#).

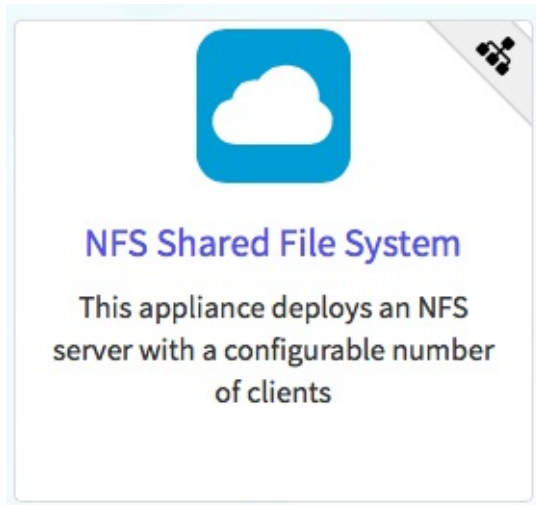


Figure 92: NFS file share

13.5.8.3 Deployment

We will explain how to launch a complex appliance based on our [NFS share appliance](#). To launch a complex appliance, you only need to follow these steps:

1. Create a lease: use the OpenStack web interface (choose between CHI@UC or CHI@TACC) to create a lease. To launch our NFS appliance, reserve at least three compute nodes (the strict minimum is two nodes but we will use three in this example and later ones).
2. Go to the [Appliance Catalog](#) and identify the appliance you want to launch. In our case you can go straight to the [NFS share appliance](#); click on it to open its details page. You will see a “Launch” button and a “Get Template” button. Follow the “Get Template” link and copy its url to the clipboard – you will need it in the following steps.
3. Click on the “Launch Complex Appliance at CHI@TACC” or “Launch Complex Appliance at CHI@UC” button depending on where your reservation was created.

This will take you to the Stacks page within the Orchestration menu. This page will show the current list of stacks, with controls to manage them and create new ones. Since we have not launched any yet, this list will be empty for now.

We will now create a new stack, which corresponds to the launch of a template. Click on Launch Stack on the top right. A window will pop up like in [Figure 93](#).

Select Template [Close]

Template Source *
File

Template File ?
Choose File no file selected

Environment Source
File

Environment File ?
Choose File no file selected

Description:
Use one of the available template source options to specify the template to be used in creating this stack.

Cancel Next

Figure 93: Select Template

We will deploy the NFS appliance described earlier; it will consist of a server node and two client nodes. Change the template source field to URL, and paste the URL of the [NFS share template](#) (if you do not have it in your clipboard anymore you will need to go back to the appliance and get it by clicking on “Get template” again).

Do not change the environment source settings, and click “Next”.

The next screen allows your to enter input values to your Heat template. Choose a name for your stack (e.g. my-nfs-cluster). Ignore the “Creation Timeout” and “Rollback On Failure” settings. You also need to enter your Chameleon password. Then, you need to select a value for the three parameters of the template: for key_name, choose your SSH key pair (this key pair will authorize access on each deployed instances, both server and client). For nfs_client_count, change the default value of 1 to 2. For reservation_id, choose your reservation created earlier. Finally, click “Launch”. As shown in [Figure 94](#).

Launch Stack



Stack Name * ?

Creation Timeout (minutes) * ?

Rollback On Failure ?

Password for user "priteau" * ?

key_name ?

nfs_client_count ?

reservation_id * ?

Description:

Create a new stack with the provided values.

Figure 94: Launch NFS Stack

Your stack should be in status “Create In Progress” for several minutes while it first launches the NFS server instance, followed by the NFS client instances. As in [Figure 95](#).

| <input type="checkbox"/> | Stack Name | Created | Updated | Status | Actions |
|--------------------------|----------------|-----------|---------|--------------------|---------------|
| <input type="checkbox"/> | my-nfs-cluster | 0 minutes | Never | Create In Progress | Check Stack ▾ |

Displaying 1 item

Figure 95: Create in Progress

It will then move to the status “Create Complete”. As the following: [Figure 96](#).

| <input type="checkbox"/> | Stack Name | Created | Updated | Status | Actions |
|--------------------------|----------------|------------|---------|-----------------|---------------|
| <input type="checkbox"/> | my-nfs-cluster | 15 minutes | Never | Create Complete | Check Stack ▾ |

Displaying 1 item

Figure 96: Create Complete

You can click on the stack name to get more details, including a visualization of the deployed resources, as pictured in [Figure 97](#). The single machine inside a circle represents the NFS server instance. The rack of machine represents the group of NFS client instances (in this case, a group composed of two instances). The server's floating IP (the public IP assigned to a resource) is represented by an IP in a circle; an IP in a circle is also used to represent the association of the IP with the NFS server instance (not the greatest idea to use the same symbol for both the IP and the association – we agree but cannot do much about it at the moment). Blow off some steam by dragging the visualization across the screen, it can be rather fun!

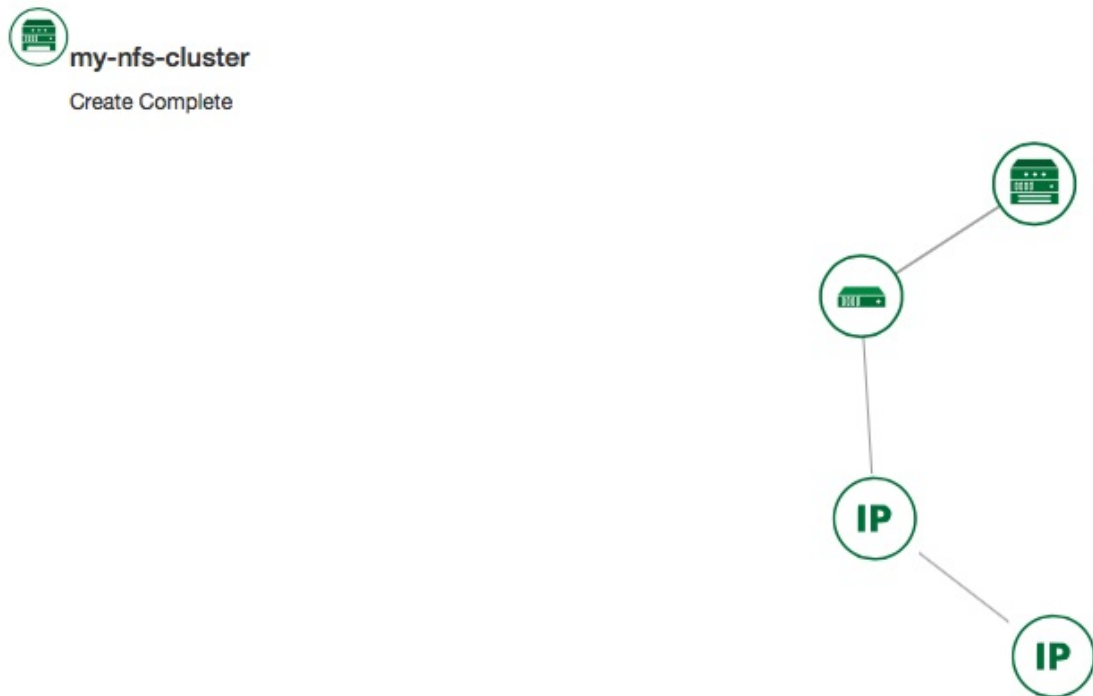


Figure 97: Stack Visualization

You can now ssh to the server using the floating IP just as you do with regular instances (use the cc account). The client does not have a floating IP attached to

it (as per the visualization given previously) but you can connect to it via the server node with the client's private IP (connect to the server with `ssh -A` to enable the SSH agent forwarding after loading your key to your SSH agent with `ssh-add <path-to-your-key>`).

You can find out the information about the IPs and other things if you click the “Overview” tab and look in the “Outputs” section. Under the “Resources” tab you will see the resources described previously (the server, clients, server's public/floating IP, and its the association) and information about them. In the “Events” tab you will see information about the history of the deployment so far. In Template you will see the template that was used to deploy this stack.

13.5.8.4 Heat Template

The NFS share appliance deploys:

- an NFS server instance, that exports the directory `/exports/example` to any instance running on Chameleon bare-metal,
- one or several NFS client instances, which configure `/etc/fstab` to mount this NFS share to `/mnt` (and can subsequently read from and write to it).

This template is reproduced further next, and includes inline comments starting with the `#` character. There are three main sections:

- resources,
- parameters,
- outputs.

The resources section is the most important part of the template: it defines which OpenStack resources to create and configure. Inside this section you can see four resources defined:

- `nfs_server_floating_ip`
- `nfs_server`
- `nfs_server_ip_association`
- `nfs_clients`

The first resource, `nfs_server_floating_ip`, creates a floating IP on the ext-net

public network. It is not attached to any instance yet.

The second resource, `nfs_server`, creates the NFS server instance (an instance is defined with the type `OS::Nova::Server` in Heat). It is a bare-metal instance (`flavor: baremetal`) using the CC-CentOS7 image and connected to the private network named `sharednet1`. We set the keypair to use the value of the parameter defined earlier, using the `get_param` function. Similarly, the reservation to use is passed to the scheduler. Finally, a user-data script is given to the instance, which configures it as an NFS server exporting `/exports/example` to Chameleon instances.

The `nfs_server_ip_association` resource associates the floating IP created earlier with the NFS server instance.

Finally, the `nfs_clients` resource defines a resource group containing instance configured to be NFS clients and mount the directory exported by the NFS server defined earlier. The IP of the NFS server is gathered using the `get_attr` function, and placed into user-data using the `str_replace` function.

Parameters all have the same data structure: each one has a name (`key_name` or `reservation_id` in this case), a data type (number or string), a comment field called `description`, optionally a default value, and a list of constraints (in this case only one per parameter). Constraints tell Heat to match a parameter to a specific type of OpenStack resource. Complex appliances on Chameleon require users to customize at least the key pair name and reservation ID, and will generally provide additional parameters to customize other properties of the cluster, such as its size, as in this example.

Outputs are declared similarly to parameters: they each have a name, an optional description, and a value. They allow to return information from the stack to the user.

```
# This describes what is deployed by this template.
description: NFS server and clients deployed with Heat on Chameleon

# This defines the minimum Heat version required by this template.
heat_template_version: 2015-10-15

# The resources section defines what OpenStack resources are to be deployed and
# how they should be configured.
resources:
  nfs_server_floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: ext-net
```



```

nfs_server:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
    key_name: { get_param: key_name }
    networks:
      - network: sharednet1
    scheduler_hints: { reservation: { get_param: reservation_id } }
    user_data: |
      #!/bin/bash
      yum install -y nfs-utils
      mkdir -p /exports/example
      chown -R cc:cc /exports
      echo '/exports/example 10.140.80.0/22(rw,async) 10.40.0.0/23(rw,async)' >> /etc/exports
      systemctl enable rpcbind && systemctl start rpcbind
      systemctl enable nfs-server && systemctl start nfs-server

nfs_server_ip_association:
  type: OS::Nova::FloatingIPAssociation
  properties:
    floating_ip: { get_resource: nfs_server_floating_ip }
    server_id: { get_resource: nfs_server }

nfs_clients:
  type: OS::Heat::ResourceGroup
  properties:
    count: { get_param: nfs_client_count }
    resource_def:
      type: OS::Nova::Server
      properties:
        flavor: baremetal
        image: CC-CentOS7
        key_name: { get_param: key_name }
        networks:
          - network: sharednet1
        scheduler_hints: { reservation: { get_param: reservation_id } }
        user_data:
          str_replace:
            template: |
              #!/bin/bash
              yum install -y nfs-utils
              echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
              mount -a
            params:
              $nfs_server_ip: { get_attr: [nfs_server, first_address] }

# The parameters section gathers configuration from the user.
parameters:
  nfs_client_count:
    type: number
    description: Number of NFS client instances
    default: 1
    constraints:
      - range: { min: 1 }
      - description: There must be at least one client.
  key_name:
    type: string
    description: Name of a KeyPair to enable SSH access to the instance
    default: default
    constraints:
      - custom_constraint: nova.keypair
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
      - custom_constraint: blazar.reservation

outputs:
  server_ip:
    description: Public IP address of the NFS server
    value: { get_attr: [nfs_server_floating_ip, ip] }
  client_ips:
    description: Private IP addresses of the NFS clients
    value: { get_attr: [nfs_clients, first_address] }

```

13.5.8.5 Customizing an existing template

Customizing an existing template is a good way to start developing your own. We will use a simpler template than the previous example to start with: it is the [Hello World complex appliance](#).

First, delete the stack you launched, because we will need all three nodes to be free. To do this, go back to the Project > Orchestration > Stacks page, select your stack, and then click on the red “Delete Stacks” button. You will be asked to confirm, so click on the blue “Delete Stacks” button. As the following picture: [Figure 98](#).

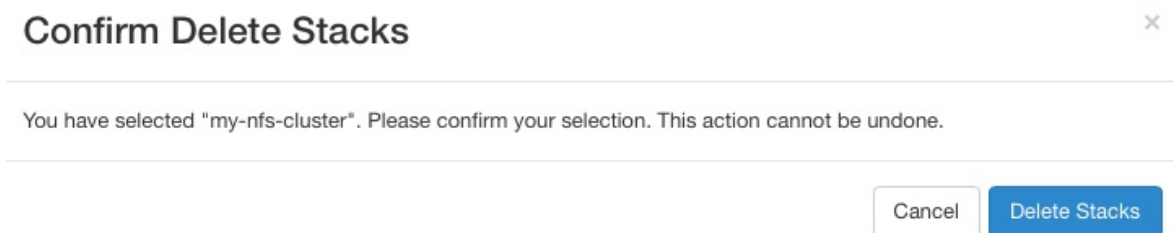


Figure 98: Delete Stacks

The template for the [Hello World complex appliance](#) is reproduced next. It is similar to the NFS share appliance, except that it deploys only a single client. You can see that it has four resources defined:

- nfs_server_floating_ip
- nfs_server
- nfs_server_ip_association
- nfs_client

The `nfs_client` instance mounts the NFS directory shared by the `nfs_server` instance, just like in our earlier example.

```
# This describes what is deployed by this template.
description: NFS server and client deployed with Heat on Chameleon

# This defines the minimum Heat version required by this template.
heat_template_version: 2015-10-15

# The resources section defines what OpenStack resources are to be deployed and
# how they should be configured.
resources:
  nfs_server_floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: ext-net
```

```
nfs_server:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
    key_name: { get_param: key_name }
    networks:
      - network: sharednet1
    scheduler_hints: { reservation: { get_param: reservation_id } }
    user_data: |
      #!/bin/bash
      yum install -y nfs-utils
      mkdir -p /exports/example
      chown -R cc:cc /exports
      echo '/exports/example 10.140.80.0/22(rw,async) 10.40.0.0/23(rw,async)' >> /etc/exports
      systemctl enable rpcbind && systemctl start rpcbind
      systemctl enable nfs-server && systemctl start nfs-server

nfs_server_ip_association:
  type: OS::Nova::FloatingIPAssociation
  properties:
    floating_ip: { get_resource: nfs_server_floating_ip }
    server_id: { get_resource: nfs_server }

nfs_client:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
    key_name: { get_param: key_name }
    networks:
      - network: sharednet1
    scheduler_hints: { reservation: { get_param: reservation_id } }
    user_data:
      str_replace:
        template: |
          #!/bin/bash
          yum install -y nfs-utils
          echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
          mount -a
        params:
          $nfs_server_ip: { get_attr: [nfs_server, first_address] }

# The parameters section gathers configuration from the user.
parameters:
  key_name:
    type: string
    description: Name of a KeyPair to enable SSH access to the instance
    default: default
    constraints:
      - custom_constraint: nova.keypair
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
      - custom_constraint: blazar.reservation
```

Download this template from the [Hello World complex appliance details page](#) to your local machine, and open it in your favorite text editor.

We will customize the template to add a second NFS client by creating a new resource called `another_nfs_client`. Add the following text to your template inside the resources section. Make sure to respect the level of indentation, which is important in YAML.

```
another_nfs_client:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
```

```

key_name: { get_param: key_name }
networks:
  - network: sharednet1
scheduler_hints: { reservation: { get_param: reservation_id } }
user_data:
  str_replace:
    template: |
      #!/bin/bash
      yum install -y nfs-utils
      echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
      mount -a
    params:
      $nfs_server_ip: { get_attr: [nfs_server, first_address] }

```

Now, launch a new stack with this template. Since the customized template is only on your computer and cannot be addressed by a URL, use the “Direct Input” method instead and copy/paste the content of the customized template. The resulting topology view is shown in [Figure 99](#), as you can see, the two client instances are shown separately since each one is defined as a separate resource in the template.

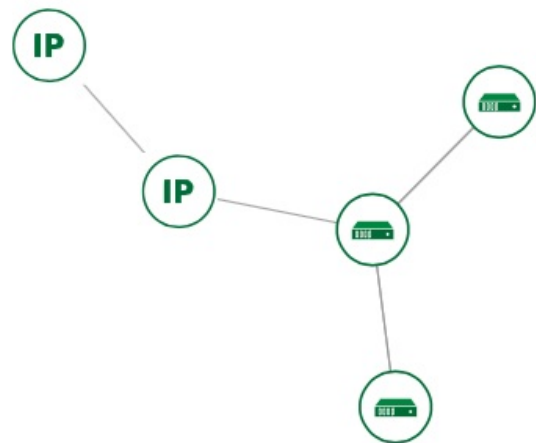


Figure 99: NFS with Two Clients

You may have realized already that while adding just one additional client instance was easy, launching more of them would require to copy / paste blocks of YAML many times while ensuring that the total count is correct. This would be easy to get wrong, especially when dealing with tens or hundreds of instances.

So instead, we leverage another construct from Heat: resource groups. Resource groups allow to define one kind of resource and request it to be created any number of times.

Remove the `nfs_client` and `another_client` resources from your customized template, and replace them with the following:

```
nfs_clients:
  type: OS::Heat::ResourceGroup
  properties:
    count: 2
    resource_def:
      type: OS::Nova::Server
      properties:
        flavor: baremetal
        image: CC-CentOS7
        key_name: { get_param: key_name }
        networks:
          - network: sharednet1
        scheduler_hints: { reservation: { get_param: reservation_id } }
        user_data:
          str_replace:
            template: |
              #!/bin/bash
              yum install -y nfs-utils
              echo "$nfs_server_ip:/exports/example /mnt/ nfs" > /etc/fstab
              mount -a
        params:
          $nfs_server_ip: { get_attr: [nfs_server, first_address] }
```

A resource group is configured with a `properties` field, containing the definition of the resource to launch (`resource_def`) and the number of resources to launch (`count`). Once launched, you will notice that the topology view groups all client instances under a single Resource Group icon. We use the same `resource_def` than when defining separate instances earlier.

Another way we can customize this template is by adding outputs to the template. Outputs allow a Heat template to return data to the user. This can be useful to return values like IP addresses or credentials that the user must know to use the system.

We will create an output returning the floating IP address used by the NFS server. We define an `outputs` section, and one output with the name `server_ip` and a description. The value of the output is gathered using the `get_attr` function which obtains the IP address of the server instance.

```
outputs:
  server_ip:
    description: Public IP address of the NFS server
    value: { get_attr: [nfs_server_floating_ip, ip] }
```

You can get outputs in the “Overview” tab of the Stack Details page. If you want

to use the command line, install `python-heatclient` and use the `heat output-list` and `heat output-show` commands, or get a full list in the information returned by `heat stack-show`.

Multiple outputs can be defined in the outputs section. Each of them needs to have a unique name. For example, we can add another output to list the private IPs assigned to client instances:

```
client_ips:
  description: Private IP addresses of the NFS clients
  value: { get_attr: [nfs_clients, first_address] }
```

The image: [Figure 100](#), shows the resulting outputs as viewed from the web interface. Of course IP addresses will be specific to each deployment.

| Outputs | |
|-------------------|--------------------------------------------------------------------------------------------|
| client_ips | Private IP address of the NFS clients
<pre>["10.140.82.20", "10.140.82.19"]</pre> |
| server_ip | Public IP address of the NFS server
<pre>130.202.88.157</pre> |

Figure 100: Outputs

Finally, we can add a new parameter to replace the hardcoded number of client instances by a value passed to the template. Add the following text to the parameters section:

```
nfs_client_count:
  type: number
  description: Number of NFS client instances
  default: 1
  constraints:
    - range: { min: 1 }
      description: There must be at least one client.
```

Inside the resource group definition, change `count: 2` to `count: { get_param: nfs_client_count }` to retrieve and use the parameter we just defined. When you launch this template, you will see that an additional parameter allows you to define the number of client instances, like in the NFS share appliance.

At this stage, we have fully recreated the NFS share appliance starting from the

Hello World one! The next section will explain how to write a new template from scratch.

13.5.8.6 Writing a new template

You may want to write a whole new template, rather than customizing an existing one. Each template should follow the same layout and be composed of the following sections:

- Heat template version
- Description
- Resources
- Parameters
- Outputs

13.5.8.6.1 Heat template version

Each Heat template has to include the `heat_template_version` key with a valid version of HOT (Heat Orchestration Template). Chameleon bare-metal supports any HOT version up to 2015-10-15, which corresponds to OpenStack Liberty. The [Heat documentation](#) lists all available versions and their features. We recommended that you always use the latest supported version to have access to all supported features:

```
heat_template_version: 2015-10-15
```

13.5.8.6.2 Description

While not mandatory, it is good practice to describe what is deployed and configured by your template. It can be on a single line:

```
description: This describes what this Heat template deploys on Chameleon.
```

If a longer description is needed, you can provide multi-line text in YAML, for example:

```
description: >
  This describes what this Heat
  template deploys on Chameleon.
```

13.5.8.6.3 Resources

The resources section is required and must contain at least one resource definition. A [complete list of resources types known to Heat](#) is available.

However, only a subset of them are supported by Chameleon, and some are limited to administrative use. We recommend that you only use:

- OS::Glance::Image
- OS::Heat::ResourceGroup
- OS::Heat::SoftwareConfig
- OS::Heat::SoftwareDeployment
- OS::Heat::SoftwareDeploymentGroup
- OS::Neutron::FloatingIP
- OS::Neutron::FloatingIPAssociation
- OS::Neutron::Port (advanced users only)
- OS::Nova::Keypair
- OS::Nova::Server

If you know of another resource that you would like to use and think it should be supported by the OpenStack services on Chameleon bare-metal, please let us know via our help desk.

13.5.8.6.4 Parameters

Parameters allow users to customize the template with necessary or optional values. For example, they can customize which Chameleon appliance they want to deploy, or which key pair to install. Default values can be provided with the `default` key, as well as constraints to ensure that only valid OpenStack resources can be selected. For example, `custom_constraint: glance.image` restricts the image selection to an available OpenStack image, while providing a pre-filled selection box in the web interface. [More details about constraints](#) are available in the Heat documentation.

13.5.8.6.5 Outputs

Outputs allow template to give information from the deployment to users. This can include usernames, passwords, IP addresses, hostnames, paths, etc. The outputs declaration is using the following format:

```
outputs:
```



```
first_output_name:  
  description: Description of the first output  
  value: first_output_value  
second_output_name:  
  description: Description of the second output  
  value: second_output_value
```

Generally values will be calls to `get_attr`, `get_param`, or some other function to get information from parameters or resources deployed by the template and return them in the proper format to the user.

13.5.8.7 Sharing new complex appliances

If you have written your own complex appliances or substantially customized an existing one, we would love if you shared them with our user community!

The process is very similar to regular appliances: log into the Chameleon portal, go to the [appliance catalog](#), and click on the button in the top-right corner: “Add an appliance” (you need to be logged in to see it as the following: [Figure 101](#)).

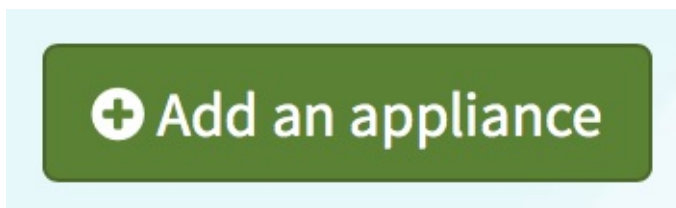


Figure 101: Add an Appliance

You will be prompted to enter a name, description, and documentation. Instead of providing appliance IDs, copy your template to the dedicated field. Finally, share your contact information and assign a version string to your appliance. Once submitted, your appliance will be reviewed. We will get in touch if a change is needed, but if it’s all good we will publish it right away!

13.5.8.8 Advanced topics

13.5.8.8.1 All-to-all information exchange

The previous examples have all used user-data scripts to provide instances with contextualization information. While it is easy to use, this contextualization method has a major drawback: because it is given to the instance as part of its launch request, it cannot use any context information that is not yet known at

this time.

In practice, this means that in a client-server deployment, only one of these patterns will be possible:

- The server has to be deployed first, and once it is deployed, the clients can be launched and contextualized with information from the server. The server will not know about the clients unless there is a mechanism (not managed by Heat) for the client to contact the server.
- The clients have to be deployed first, and once they are deployed, the server can be launched and contextualized with information from the clients. The clients will not know about the server unless there is a mechanism (not managed by Heat) for the server to contact the clients.

This limitation was already apparent in our NFS share appliance: this is why the server instance exports the file system to all bare-metal instances on Chameleon, because it does not know which specific IP addresses are allocated to the clients.

This limitation is even more important if the deployment is not hierarchical, i.e. all instances need to know about all others. For example, a cluster with IP and hostnames populated in `/etc/hosts` required each instance to be known by every other instance.

This section presents a more advanced form of contextualization that can perform this kind of information exchange. This is implemented by Heat agents running inside instances and communicating with the Heat service to send and receive information. This means you will need to use an image bundling these agents. Currently, our `CC-CentOS7` appliance and its `CUDA` version are the only ones supporting this mode of contextualization. If you build your own images using the [CC-CentOS7 appliance builder](#), you will automatically have these agents installed.

This contextualization is performed with several Heat resources:

- `OS::Heat::SoftwareConfig`. This resource describes code to run on an instance. It can be configured with inputs and provide outputs.
- `OS::Heat::SoftwareDeployment`. This resource applies a `SoftwareConfig` to a specific

instance.

- `OS::Heat::SoftwareDeploymentGroup`. This resource applies a `SoftwareConfig` to a specific group of instances.

The template next illustrates how it works. It launches a group of instances that will automatically populate their `/etc/hosts` file with IP and hostnames from other instances in the deployment.

```
heat_template_version: 2015-10-15

description: >
  This template demonstrates how to exchange hostnames and IP addresses to populate /etc/hosts.

parameters:
  flavor:
    type: string
    default: baremetal
    constraints:
      - custom_constraint: nova.flavor
  image:
    type: string
    default: CC-CentOS7
    constraints:
      - custom_constraint: glance.image
  key_name:
    type: string
    default: default
    constraints:
      - custom_constraint: nova.keypair
  instance_count:
    type: number
    default: 2
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
      - custom_constraint: blazar.reservation

resources:
  export_hosts:
    type: OS::Heat::SoftwareConfig
    properties:
      outputs:
        - name: hosts
      group: script
      config: |
        #!/bin/sh
        (echo -n $(factor ipaddress); echo -n ' '; echo $(factor hostname)) > ${heat_outputs_path}.hosts

  export_hosts_sdg:
    type: OS::Heat::SoftwareDeploymentGroup
    properties:
      config: { get_resource: export_hosts }
      servers: { get_attr: [server_group, refs_map] }
      signal_transport: HEAT_SIGNAL

  populate_hosts:
    type: OS::Heat::SoftwareConfig
    properties:
      inputs:
        - name: hosts
      group: script
      config: |
        #!/usr/bin/env python
        import ast
        import os
        import string
        import subprocess
        hosts = os.getenv('hosts')
        if hosts is not None:
            hosts = ast.literal_eval(string.replace(hosts, '\n', '\\n'))
```

```

with open('/etc/hosts', 'a') as hosts_file:
    for ip_host in hosts.values():
        hosts_file.write(ip_host.rstrip() + '\n')

populate_hosts_sdg:
  type: OS::Heat::SoftwareDeploymentGroup
  depends_on: export_hosts_sdg
  properties:
    config: { get_resource: populate_hosts }
    servers: { get_attr: [server_group, refs_map] }
    signal_transport: HEAT_SIGNAL
    input_values:
      hosts: { get_attr: [ export_hosts_sdg, hosts ] }

server_group:
  type: OS::Heat::ResourceGroup
  properties:
    count: { get_param: instance_count }
    resource_def:
      type: OS::Nova::Server
      properties:
        flavor: { get_param: flavor }
        image: { get_param: image }
        key_name: { get_param: key_name }
        networks:
          - network: sharednet1
        scheduler_hints: { reservation: { get_param: reservation_id } }
        user_data_format: SOFTWARE_CONFIG
        software_config_transport: POLL_SERVER_HEAT

outputs:
  deployment_results:
    value: { get_attr: [export_hosts_sdg, hosts] }

```

There are two SoftwareConfig resources.

The first SoftwareConfig, `export_hosts`, uses the `factor` tool to extract IP address and hostname into a single line (in the format expected for `/etc/hosts`) and writes it to a special path (`/${heat_outputs_path}.hosts`). This prompts Heat to assign the content of this file to the output with the name `hosts`.

The second SoftwareConfig, `populate_hosts`, takes as input a variable named `hosts`, and applies a script that reads the variable from the environment, parses it with `ast.literal_eval` (as it is formatted as a Python dict), and writes each value of the dictionary to `/etc/hosts`.

The `SoftwareDeploymentGroup` resources `export_hosts_sdg` and `populate_hosts_sdg` apply each SoftwareConfig to the instance ResourceGroup with the correct configuration.

Finally, the instance ResourceGroup is configured so that each instance uses the following contextualization method instead of a user-data script:

```

user_data_format: SOFTWARE_CONFIG
software_config_transport: POLL_SERVER_HEAT

```

You can follow the same template pattern to configure your own deployment

requiring all-to-all information exchange.

13.5.9 Openstack Bare Metal [🌸](#)

In this page you will find documentation guiding you through the bare-metal deployment features available in Chameleon. Chameleon gives users administrative access to bare-metal compute resources to run cloud computing experiments with a high degree of customization and repeatability. Typically, an experiment will go through several phases, as illustrated in [+Figure 102](#).

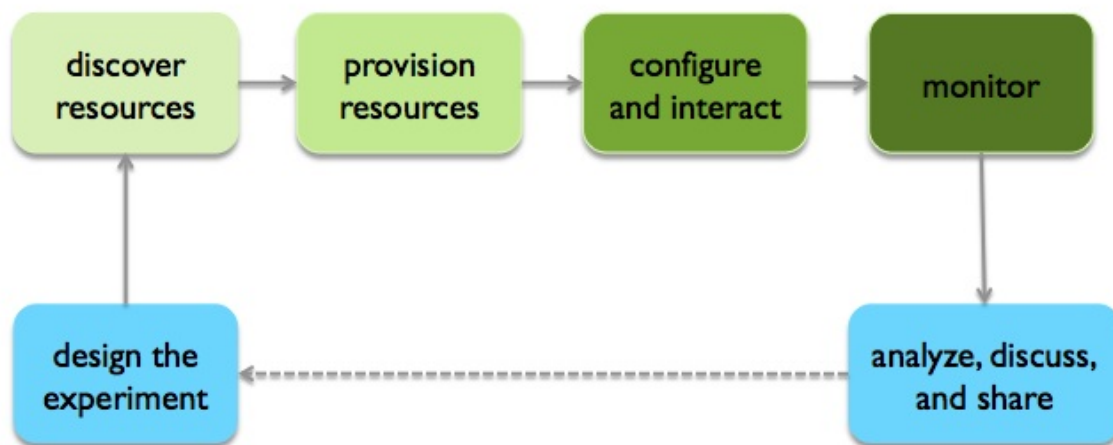


Figure 102: Experimenting with Bare Metal resources on Chameleon cloud

The bare-metal user guide comes in two editions. The first is how to use Chameleon resources via the web interface, the recommended choice for new users to quickly learn how to use our testbed:

[Get started with Chameleon using the web interface](#)

1. [Discover Resources](#)
2. [Provision Resources](#)
3. [Configure and Interact](#)
4. [Monitor and Collect Results](#)

The second targets advanced users who are already familiar with Chameleon and would like to learn how to use Chameleon from the command line or with scripts.

[Get started with Chameleon using the command line \(advanced\)](#)

1. [Discover Resources](#)
2. [Provision Resources](#)
3. [Configure and Interact](#)
4. [Monitor and Collect Results](#)

You do not need to strictly follow the documentation sequentially. However, note that some steps assume that previous ones have been successfully performed.

You can also consult documentation describing how to use advanced features of Chameleon not covered by the previous guides:

- the [Chameleon Object Store](#),
- [network isolation for bare metal](#).

13.5.10 Chameleon Cloud Frequently Asked Questions

13.5.10.1 Appliances

13.5.10.1.1 What is an appliance?

An appliance is an application packaged together with the environment that this application requires. For example, an appliance can consists of the operating system, libraries and tools used by the application, configuration features such as environment variable settings, and the installation of the application itself. Examples of appliances might include a KVM virtual machine image, a Docker image, or a bare metal image. Chameleon appliance refers to bare metal images that can be deployed on the Chameleon testbed. Since an appliance captures the

experimental environment exactly, it is a key element of reproducibility; publishing an appliance used to obtain experimental results will go a long way to allowing others to reproduce and build on your research easily.

To deploy distributed applications on several Chameleon instances, complex appliances combine an image and a template describing how the cluster should be configured and contextualized. You can read more about them in the [Complex Appliances documentation](#).

13.5.10.1.2 What is the Appliance Catalog?

[The Chameleon Appliance Catalog](#) is a repository that allows users to discover, publish, and share appliances. The appliance catalog contains useful images of both bare metal and virtual machine appliances supported by the Chameleon team as well appliances contributed by users.

13.5.10.1.3 How do I publish an appliance in the Appliance Catalog?

The new Appliance Catalog allows you to easily publish and share your own appliances so that others can discover them and use them either to reproduce the research of others or as a basis for their own research. Before creating your own appliance it is advisable to review other appliances on the [Chameleon Appliance Catalog](#) in order to get an idea of the categories you will want to contribute and what others have done.

Once you are ready to proceed, an appliance can be contributed to Chameleon in the following steps:

1. Create the appliance itself. You may want to test it as well as give some thought to what support you are willing to provide for the appliance (e.g., if your group developed and supports a software package, the appliance may be just a new way of packaging the software and making it available, in which case your standard support channels may be appropriate for the appliance as well).
2. Upload the appliance to the Chameleon Image Repository (Glance) and make the image public. In order to enter the appliance into the Catalog you will be asked to provide the Glance ID for the image. These IDs are per-

cloud, so that there are three options right now: bare metal/CHI at University of Chicago, bare metal/CHI at TACC, and OpenStack/KVM at TACC. You will need to provide at least one appliance, but may want to provide all three.

3. Go to the [Appliance Catalog Create Appliance web form](#), fill out, and submit the form. Be prepared to provide the following information: a descriptive name (this sometimes requires some thought!), author and support contact, version, and an informative description. The description is a very important part of the appliance record; others will use it to evaluate if the appliance contains tools they need for their research so it makes sense to prepare it carefully. To make your description effective you may want to think of the following questions: what does the appliance contain? what are the specific packages and their versions? what is it useful for? where can it be deployed and/or what restrictions/limitations does it have? how should users connect to it / what accounts are enabled?

If you are adding a complex appliance, skip the image ID fields and enter your template instead in the dedicated text box.

As always, if you encounter any problems or want to share with us additional improvements we should do to the process, please do not hesitate to [submit a ticket](#).

13.5.10.1.4 How can I manage an appliance on Appliance Catalog?

If you are the owner of the appliance, you can edit the appliance data, such as the description or the support information. Browse to the appliance that you want to edit and view its Details page. At the top right of the page is an Edit button. You will be presented with the same web form as when creating the appliance, pre-filled with the appliances current information. Make changes as necessary and click Save at the bottom of the page.

And finally, you can delete appliances you had made available. Browse to the appliance that you want to delete and click Edit on the Appliance Details page. At the bottom of the page is a Delete button. You will be asked to confirm once more that you do want to delete this appliance. After confirming, the appliance will be removed and no longer listed on the Appliance Catalog.

13.5.10.1.5 Why are there different image IDs for the same appliance?

The three clouds forming the Chameleon testbed are fully separated, each having its own Glance image repository. The same appliance image uploaded to the three clouds will produce three different image IDs.

In addition, it is sometimes needed to customize an appliance image for each site, resulting in slightly different image files.

13.5.10.1.6 Can I use another operating system on bare-metal?

The recommended appliance for Chameleon is CentOS 7 (supported by Chameleon staff), or appliances built on top of it.

These appliances provide Chameleon-specific customizations, such as login using the cc account, the cc-checks utility to verify hardware against our resource registry, gathering of metrics, etc.

Since 2016, we also provide and support Ubuntu 14.04 and 16.04 appliances with the same functionality.

13.5.10.2 Bare Metal Troubleshooting

13.5.10.2.1 Why are my Bare Metal instances failing to launch?

The Chameleon Bare Metal clouds require users to reserve resources before allowing them to launch instances. Please follow the [documentation](#) and make sure that:

1. You have created a lease and it has started (the associated reservation is shown as **Active**)
2. You have selected your reservation in the **Launch Instance** panel

If you still cannot start instances, please [open a ticket with our help desk](#).

13.5.10.3 OpenStack KVM Troubleshooting

13.5.10.3.1 Why are my OpenStack KVM instances failing to launch?

If you get an error stating that **No valid host was found**, it might be caused by a lack of resources in the cloud. The Chameleon staff continuously monitors the utilization of the testbed, but there might be times when no more resources are available. If the error persists, please [open a ticket with our help desk](#).

13.5.10.3.2 Why can I not ping or SSH to my instance?

While the possibility that the system is being taken over by nanites should not be discounted too easily, it is always prudent to first check for the following three settings:

- Do you have a floating IP associated with your instance? By default, instances do not have publicly-accessible IP addresses assigned. See our documentation on [Associating a Floating IP Address](#).
- Does your security group allow incoming ICMP (e.g. ping) traffic? By default, firewall rules do not allow ping to your instances. If you wish to enable it, see our documentation on [Adding a Security Group to an Instance](#).
- Does your security group allow incoming SSH (TCP port 22) traffic? By default, firewall rules do not allow SSH to your instances. If you wish to enable it, see our documentation on [Adding a Security Group to an Instance](#).

If none of these solve your problem, please [open a ticket with our help desk](#), and send us the results of the three previous settings (and any evidence of nanites you find as well).

13.6 GLOSSARY

13.6.1 Glossary



13.6.1.1 VM and Container

Dom0

TBD

Hypervisor

TBD

KVM

TBD

Virtual Machine

TBD

Virtual Machine Manager

TBD

XEN

TBD

cgroups

TBD

chroot

TBD

container

TBD

Kernel namespace

TBD

13.6.1.2 Network

Bridged Networking

TBD

External Network

TBD

Internal Network

TBD

Local Bridge

TBD

Network Address Translation (NAT)

TBD

13.6.1.3 Storage

Block Device

TBD

Virtual Disk

TBD

Raw Disk

TBD

13.6.2 Creating the ePubs from source

In case you wish to create the ePub from source, we have included this section.

The creation of the book is based on [bookmanager](#).

The easiest way is to use our docker container as described in [Section 13.6.2.1](#).

13.6.2.1 Docker

We recommend the docker creation method for

- Ubuntu
- Windows 10
- macOSX

13.6.2.1.1 Using OSX

The easiest way to create a system that can compile the book on macOS, is to use a docker container. To do so you will need to first install docker on macOS while following the simple instructions at

- <https://docs.docker.com/docker-for-mac/install/>

Once you have docker installed, you can follow the instructions in [Section 13.6.2.1](#).

13.6.2.1.2 Using the Docker Image

In case you have docker installed on your computer you can create ePubs with our docker image. To create that image by hand, we have included a simple makefile. Alternatively you can use our image from dockerhub if you like, it is based on ubuntu and uses our [Dockerfile](#).

First, you need to download the repository:

```
$ git clone https://github.com/cloudmesh-community/book.git
cd book
```

To open an interactive shell into the image you say

```
$ make shell
```

The container image includes

- [Python 3.7.4](#)
- [Pandoc 2.7.3](#)
- [pandoc-citeproc](#)

Now you can skip to [Section 13.6.2.4](#) and compile the book just as documented there.

Please note that we have not integrated `pandoc-mermaid` and `pandoc-index` at this time in our docker image. If you like to contribute them, please try it and make a pull request once you got them to work.

In case you want to create or recreate the image from our [Dockerfile](#) (which is likely not necessary, you can use the command

```
$ make image
```

13.6.2.2 Using the Native System

In case you like to use your native environment (which is typically faster than the container) you need to make sure you have an up to date environment.

Please note, that you must have at least Pandoc version 2.5 installed as earlier versions will not work. We recommend that you use pandoc version 2.7.3 or newer. We recommend that you use Python version 3.7.4 to run the scripts needed to assemble the document. However earlier version of Python 3 may also work, but are not tested. You can check the versions with

```
$ pandoc --version  
$ python --version
```

13.6.2.3 Using Vagrant

In case you have installed vagrant on your computer which is available for macOS, Linux, and Windows 10, you can use our vagrant file to start up a virtual machine that has all software installed to create the ePub.

First, you need to download the repository:

```
$ git clone https://github.com/cloudmesh-community/book.git  
$ cd book
```

Next you have to create the virtual machine with

```
$ vagrant up
```

You can log into the VM with

```
$ vagrant ssh
```

The book folder will be mounted in the VM and you can follow the instructions in [Section 13.6.2.1](#).

13.6.2.4 Creating a Book

Once you have decided for one of the methods, you can create a book.

To create a book, you have to first check out the book source from github with if you have not yet done so (for example if you were to use the docker container method):

```
git clone git@github.com:cloudmesh-community/book.git
```

Books are organized in directories. We currently have created the following directories

```
./book/books/cloud/  
./book/books/big-data-applications/  
./book/books/pi  
./book/books/writing  
./book/books/222  
./book/books/516
```

To compile a book go to the directory and make it. Let us assume you like to create the cloud book for cloud

```
$ git clone https://github.com/cloudmesh-community/book.git  
$ cd book/books/cloud  
$ make
```

To view it you say


```
$ make view
```

After you have done modifications, you need to do one of two things. In case you add new images you need to use

```
$ make
```

The structure of the books is maintained in the yml file in the directory where you execute the make in. It typically has the form `NAMEOFDIR.yml`. Simply do an `ls` in the directory to see its name or inspect the Makefile. You can add new chapters to the yml file, but discuss this first with Gregor. Typically, we have for incoming or draft chapters a special `draft` book to make sure the integration is done smoothly first in the draft.

13.6.2.5 Publishing the Book to GitHub

 *This task is only to be done by Gregor von Laszewski. You will not have to do this step.*

To publish the book say

```
$ make publish
```

13.6.2.6 Creating Drafts

Drafts are maintained in the draft folder

```
$ cd book/books/cloud  
$ make
```

We recommend that you use the following tools to clean up your files.

- [mdl](#) - markdownlint to cleanup your markdown
- [biber](#) - to cleanup your bibtex file

We still only use bibtex and not biblatex, but can use biber for doing some verification. Once you have installed them, you can verify your documents with.

```
mdl filename.md  
biber -V -tool filename.bib
```

Please remember that we have many thousands of references in our bib folder, so before you add a duplicate entry, please check in that folder. An easy way to do this is to use `jabref` loading the bibfiles.

13.6.2.7 Creating a New Book

Let us assume you like to create a new book. The easiest way to start is to copy from an existing book. However, make sure not to copy old files in dest. Let us assume you like to call the book gregor and you copy from the python directory.

You have to do the following

```
$ cd book/books/python
$ make clean
$ cd ..
$ cp -r python gregor
$ cd ../gregor
$ mv python.yaml gregor.yaml
```

edit the Makefile and replace the NAME with gregor. make modifications to the table of contents in that yaml file and then compile with

```
$ make
```

13.6.2.8 Managing Images

In case you have added images to the book, they must be on the same level as your contribution, but in a directory called images. E.g.

```
./chapters/cloud/mydocument.md
./chapters/cloud/images/myimage.md
```

In the document the image is then referred to as

```
![My image caption](images/myimage.md){#fig:cloud-myimage}
```

The label `#fig:cloud-myimage` must be unique in all of the documents. While adding the directory cloud before the image name this is the case in our example.

13.6.2.9 Managing References

References are all managed in bibtex format while using pandoc-croref to cite them. There are many examples of the different entry types available in the bib directory. DO not duplicate entries, instead reuse them. Make sure you have a unique and meaningful label.

14 REFERENCES



- [1] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah, “The julia language.” Website, Sep-2019 [Online]. Available: <https://www.julialang.org>
- [2] K. Finley, “Out in the open: Man creates one programming language to rule them all.” Magazine, Feb-2014 [Online]. Available: <https://www.wired.com/2014/02/julia/>
- [3] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah, “Julia: A fresh approach to numerical computing,” *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017 [Online]. Available: <https://doi.org/10.1137/141000671>
- [4] I. Balbaert, *Getting started with julia programming language*. Packt Publishing - ebooks Account, 2015 [Online]. Available: <https://www.xarg.org/ref/a/178328479X/>
- [5] K. Sato, “Docopt: A command line argument parser for julia.” Code Repository, Oct-2018 [Online]. Available: <https://github.com/docopt/DocOpt.jl>
- [6] S. G. Johnson, “Package to call python functions from the julia language.” Code Repository, Aug-2019 [Online]. Available: <https://github.com/JuliaPy/PyCall.jl>
- [7] A. Sengupta, “Call java from julia: JavaCall.” Code Repository, Jun-2019 [Online]. Available: <http://juliainterop.github.io/JavaCall.jl>
- [8] T. Mohapatra, “Hadoop hdfs and yarn client: Elly.jl.” Code Repository, Jul-2018 [Online]. Available: <https://github.com/JuliaParallel/Elly.jl>
- [9] S. O’Connor, “Julia awscore: Amazon web services core functions and types.” Code Repository, Aug-2019 [Online]. Available: <https://github.com/JuliaCloud/AWSCore.jl>
- [10] É. P. Fédérale, “Unified-types.” Web Page, Feb-2019 [Online]. Available:

<https://docs.scala-lang.org/tour/unified-types.html>

[11] C. Artho, K. Havelund, R. Kumar, and Y. Yamagata, “Domain-specific languages with scala,” in *International conference on formal engineering methods*, 2015, pp. 1–16.

[12] AWS, “Amazon aurora – relational database built for the cloud.” Web Page, Aug-2019 [Online]. Available: <https://aws.amazon.com/rds/aurora/>

[13] Amazon, “Amazon aws ml products and services.” Web Page, 2019 [Online]. Available: <https://aws.amazon.com/machine-learning/>

[14] Google, “Google cloud ai and ml products.” Web Page, 2019 [Online]. Available: <https://cloud.google.com/products/ai/>

[15] “IBM watson product page.” Web Page, Jan-2017 [Online]. Available: <https://www.ibm.com/watson>

[16] Microsoft, “Microsoft azure ml products.” Web Page, 2019 [Online]. Available: <https://studio.azureml.net/>

[17] Google, “Google cloud vision overview.” Web Page, 2019 [Online]. Available: <https://cloud.google.com/vision/>

[18] Google, “Requests to google cloud vision api.” Web Page, 2019 [Online]. Available: <https://cloud.google.com/vision/docs/request>

[19] scikit-learn, “Scikit-learn feature extraction algorithm.” Web Page, 2019 [Online]. Available: https://scikit-learn.org/stable/modules/feature_extraction.html

[20] M. Wittig Andreas; Wittig, *Amazon web services in action*, 1st ed. Manning Press, 2015.

[21] Y. Brikman, *Terraform: Up and running*, 1st ed. O’Reilly Media Inc, 2017.

[22] K. Morris, *Infrastructure as code*, 1st ed. O’Reilly Media Inc, 2015.

[23] AWS, “AWS elastic beanstalk – deploy web applications.” Web Page, Aug-

2019 [Online]. Available: <https://aws.amazon.com/elasticbeanstalk/>