

CLOUD COMPUTING

Gregor von Laszewski

Fugang Wang

Geoffrey C. Fox

laszewski@gmail.com

CLOUD COMPUTING

Gregor von Laszewski

(c) Gregor von Laszewski, 2018

CLOUD COMPUTING

1. PREFACE	•
1.1 VERSION	•
1.2 EPUB READERS	•
1.3 CORRECTIONS	•
1.4 CONTRIBUTORS	•
1.5 CREATING THE EPUBS FROM SOURCE	•
1.5.1 Ubuntu requirements	○
1.5.2 Using Windows 10	
1.5.3 Using Vagrant	
1.5.4 Using OSX	
1.5.5 Docker	
1.5.6 Creating a book	
1.5.7 Publishing the book to github	
1.5.8 Creating Drafts that are not yet published	
1.5.9 Creating a new book	
1.6 NOTATION	•
1.6.1 Hyperlinks in the document	
1.6.2 Equations	
1.7 UPDATES	•
1.8 WEEKLY ACTIVITIES	•
1.8.1 Week 1: Activities Jan 10 - 17	
1.8.2 Week 2: Jan 17 - 25	
1.8.3 Week 3: Jan 25 - Feb 1	
1.8.4 Week 4: Feb 1 - Feb 8	
2. GITBUB Book Issues	•
2.1 GITHUB ISSUES	•
2.1.1 Internal Issues	
2.1.2 Open Sections	
2.1.3 Open Chapters	
2.1.4 Assigned Sections	
2.1.5 Assigned Chapters	
2.1.6 Open Projects	
2.1.7 Assigned Projects	
3. QUICK START	•

<u>3.1 HELP</u>	✿
<u>3.2 How to take this class</u>	✿
<u>3.3 ASSIGNMENTS</u>	✿
<u>3.3.1 Account Creation</u>	
<u>3.3.2 Sections, Chapters with Examples</u>	
<u>3.3.3 Mini Projects that could Substitute a Chapter</u>	
<u>3.3.4 Project</u>	
<u>3.3.5 Project: Virtual Cluster</u>	
<u>3.3.6 Submission of sections and chapters and projects</u>	
<u>3.3.7 Participation</u>	
<u>3.4 COURSE SYLLABUS TABLES</u>	✿
<u>3.4.1 Proposed Lecture Timeline</u>	
<u>3.4.2 Proposed Assignments Timeline</u>	
<u>3.4.3 Group Breakdown Checkpoint</u>	
<u>4 CLASS OVERVIEW</u>	✿
<u>4.1 ORGANIZATION</u>	✿
<u>4.1.1 First Week</u>	
<u>4.1.2 IoT Hardware</u>	
<u>4.1.3 Access to Clouds</u>	
<u>4.1.4 Using Your Own Computer</u>	
<u>4.1.5 Parallel Tracks</u>	
<u>4.1.6 Plagiarism</u>	✿
<u>4.2 RESEARCH INTERESTS</u>	✿
<u>4.3 E516: ENGINEERING CLOUD COMPUTING</u>	✿
<u>4.3.1 Course Description</u>	
<u>4.3.2 Course Objectives</u>	
<u>4.3.3 Learning Outcomes</u>	
<u>4.4 Syllabus</u>	
<u>4.4.1 Assessment</u>	
<u>4.5 COURSE POLICIES</u>	✿
<u>4.5.1 Discussion via Piazza</u>	
<u>4.5.2 Managing Your Own Calendar</u>	
<u>4.5.3 Online and Office Hours</u>	
<u>4.5.4 Class Material</u>	
<u>4.5.5 HID</u>	
<u>4.5.6 Class Directory</u>	
<u>4.5.7 Notebook</u>	

[4.5.8 Blog](#)

[4.5.9 Waitlist](#)

[4.5.10 Registration](#)

[4.5.11 Auditing the class](#)

[4.5.12 Resource restrictions](#)

[4.5.13 Incomplete](#)

[4.6 E516 SUMMARY](#)

[4.6.1 Introduction](#)

[4.6.2 Class communication](#)

[4.6.3 Assignments](#)

[4.6.4 Scientific Writing](#)

[4.7 EXAMPLE ARTIFACTS](#)

[4.7.1 Technology Summaries](#)

[4.7.2 Chapters](#)

[4.7.3 Project Reports](#)

[5 DEFINITION OF CLOUD COMPUTING](#)

[5.1 Defining the term Cloud Computing](#)

[5.2 History and Trends](#)

[5.3 Job as a Data Engineer](#)

[5.4 You must be that TALLL](#)

[6 DATA CENTER](#)

[6.1 Motivation: Data](#)

[6.1.1 How much data?](#)

[6.2 Cloud Data Centers](#)

[6.3 Data Center Infrastructure](#)

[6.4 Data Center Characteristics](#)

[6.5 Data Center Metrics](#)

[6.5.1 Data Center Carbon Footprint](#)

[6.5.2 Data Center Operational Impact](#)

[6.5.3 Power Usage Effectiveness](#)

[6.5.4 Hot-Cold Aisle](#) 

[6.5.5 Workload Monitoring](#)

[6.6 Example Data Centers](#)

[6.6.1 AWS](#)

[6.6.2 Azure](#)

[6.6.3 Google](#)

[6.6.4 IBM](#)

- [6.6.5 XSEDE](#)
- [6.6.6 Chameleon Cloud](#)
- [6.6.7 Indiana University](#)
- [6.6.8 Shipping Containers](#)
- [6.7 Server Consolidation](#)
- [6.8 Data Center Improvements and Consolidation](#)
- [6.9 Project Natick](#)
- [6.10 Renewable Energy for Data Centers ○](#)
- [6.11 Societal Shift Towards Renewables ○](#)
- [6.12 Exercises](#)

7 ARCHITECTURES



- [7.1 Evolution of Compute Architectures](#)
- [7.1.1 Mainframe Computing](#)
- [7.1.2 PC Computing](#)
- [7.1.3 Intranet and Server Computing](#)
- [7.1.4 Grid Computing Computing](#)
- [7.1.5 Internet Computing](#)
- [7.1.6 Cloud Computing](#)
- [7.1.7 Mobile Computing](#)
- [7.1.8 Internet of Things Computing](#)
- [7.1.9 Edge Computing](#)
- [7.1.10 Fog Computing](#)
- [7.2 As a Service Architecture Model](#)
- [7.3 Product or Functional Based Model](#)
- [7.4 NIST Cloud Architecture](#)
- [7.5 Cloud Security Alliance Reference Architecture](#)
- [7.6 Multicloud Architectures](#)

- [7.6.1 Cloudmesh Architecture](#)

- [7.7 Resources](#)

8 NIST BIG DATA REFERENCE ARCHITECTURE



- [8.1 Pathway to the NIST-BDRA](#)
- [8.2 Big Data Characteristics and Definitions](#)
- [8.3 Big Data and the Cloud](#)
- [8.4 Big Data, Edge Computing and the Cloud](#)
- [8.5 Reference Architecture](#)
- [8.6 Framework Providers](#)
- [8.7 Application Providers](#)

[8.8 Fabric](#)

[8.9 Interface definitions](#)

[9 REST](#)

[9.1 Overview of REST](#)

[9.1.1 Collection of Resources](#)

[9.1.2 Single Resource](#)

[9.1.3 REST Tool Classification](#)

[9.2 OPENAPI REST SERVICES WITH SWAGGER](#)

[9.2.1 Swagger Tools](#)

[9.2.2 Swagger Community Tools](#)

[9.3 OPENAPI SPECIFICATION](#)

[9.3.1 The Virtual Cluster example API Definition](#)

[9.3.2 References](#)

[9.4 OPENAPI REST SERVICE VIA INTROSPECTION](#)

[9.4.1 Exercise](#)

[9.5 OPENAPI REST SERVICE VIA CODEGEN](#)

[9.5.1 Step 1: Define Your REST Service](#)

[9.5.2 Step 2: Server Side Stub Code Generation and Implementation](#)

[9.5.3 Step 3: Install and Run the REST Service:](#)

[9.5.4 Step 4: Generate Client Side Code and Verify](#)

[9.5.5 Towards a Distributed Client Server](#)

[9.5.6 Exercises](#)

[9.6 FLASK RESTFUL SERVICES](#)

[9.7 REST SERVICES WITH EVE](#)

[9.7.1 Ubuntu install of MongoDB](#)

[9.7.2 macOS install of MongoDB](#)

[9.7.3 Windows 10 Installation of MongoDB](#)

[9.7.4 Database Location](#)

[9.7.5 Verification](#)

[9.7.6 Building a simple REST Service](#)

[9.7.7 Interacting with the REST service](#)

[9.7.8 Creating REST API Endpoints](#)

[9.7.9 REST API Output Formats and Request Processing](#)

[9.7.10 REST API Using a Client Application](#)

[9.7.11 Towards cmd5 extensions to manage eve and mongo](#) O

[9.8 HATEOAS](#)

9.8.1 Filtering	⊕
9.8.2 Pretty Printing	⊕
9.8.3 XML	⊕
9.9 EXTENSIONS TO EVE	⊕
9.9.1 Object Management with Eve and Evegenie	⊕
9.10 DJANGO REST FRAMEWORK	⊕
9.11 GITHUB REST SERVICES	⊕
9.11.1 Issues	⊕
9.11.2 Exercise	⊕
10 VIRTUALIZATION	⊕
10.1 Virtual Machines	⊕
10.2 System Virtual Machines	⊕
10.3 Hosted Virtualization	⊕
10.4 Summary	⊕
10.5 Virtualization Approches	⊕
10.5.1 Full virtualization	⊕
10.5.2 Paravirtualization	⊕
10.6 Virtualization Technologies	⊕
10.6.1 Selected Hardware Virtualization Technologies	⊕
10.6.2 AMD-V and Intel-VT	⊕
10.6.3 I/O MMU virtualization (AMD-Vi and Intel VT-d)	⊕
10.6.4 Selected VM Virtualization Software and Tools	⊕
10.6.5 Parallels	⊕
10.6.6 Selected Storage Virtualization Software and Tools	⊕
10.6.7 Selected Network Virtualization Software and Tools	⊕
10.7 VIRTUAL MACHINE MANAGEMENT WITH QEMU	⊕
10.7.1 Install QEMU	⊕
10.7.2 Create a Virtual Hard Disk with QEMU	⊕
10.7.3 Install Ubuntu on the Virtual Hard Disk	⊕
10.7.4 Start Ubuntu with QEMU	⊕
10.7.5 Emulate Raspberry Pi with QEMU	⊕
10.7.6 Resources	⊕
10.8 MANAGE VM GUESTS WITH VIRSH ?	⊕
11 INFRASTRUCTURE AS A SERVICE	⊕
11.1 INTRODUCTION	⊕
11.2 AMAZON WEB SERVICES	⊕
11.2.1 AWS Products	⊕

[11.2.2 Locations](#)
[11.2.3 Creating an account](#)
[11.2.4 AWS Command Line Interface](#)
[11.2.5 AWS Admin Access](#)
[11.2.6 Understanding the free tier](#)
[11.2.7 Important Notes](#)
[11.2.8 Introduction to the AWS console](#)
[11.2.9 Access from the Command Line](#)
[11.2.10 Access from Python](#)
[11.2.11 Boto](#)
[11.2.12 libcloud](#)

[11.3 MICROSOFT AZURE](#)

[11.3.1 Products](#)
[11.3.2 Registration](#)
[11.3.3 Introduction to the Azure Portal](#)
[11.3.4 Creating a VM](#)
[11.3.5 Starting a VM](#)
[11.3.6 Stopping the VM](#)
[11.3.7 Exercises](#)

[11.4 WHAT IS IBM WATSON AND WHY IS IT IMPORTANT?](#)

[11.4.1 How can we use Watson?](#)
[11.4.2 Creating an account](#)
[11.4.3 Understanding the free tier](#)

[11.5 GOOGLE IAAS CLOUD SERVICES O ?](#)

[11.5.1 Cloud Computing Services and Products](#)

[11.6 OPENSTACK](#)

[11.6.1 Introduction](#)
[11.6.2 OpenStack Architecture](#)
[11.6.3 Components](#)
[11.6.4 Core Services](#)
[11.6.5 Access from Python and Scripts](#)

[11.7 PYTHON LIBCLOUD](#)

[11.7.1 Service categories](#)
[11.7.2 Installation](#)
[11.7.3 Quick Example](#)
[11.7.4 Working with cloud services](#)
[11.7.5 Cloudmesh Community Program to Manage Clouds](#)

[11.7.6 Amazon Simple Storage Service S3 via libcloud](#) 

[**11.8 AWS Boto**](#) 

[11.8.1 Boto versions](#)

[11.8.2 Boto Installation](#)

[11.8.3 Access key](#)

[11.8.4 BOTO configuration](#)

[11.8.5 EC2 interface of Boto](#)

[11.8.6 List EC2 instances](#)

[11.8.7 Amazon S3 interface of Boto](#)

[11.8.8 References](#)

[11.8.9 Exercises](#)

[**12 MAPREDUCE**](#) 

[**12.1 INTRODUCTION TO MAPREDUCE**](#) 

[12.1.1 MapReduce Algorithm](#)

[12.1.2 Hadoop MapReduce and Hadoop Spark](#)

[12.1.3 References](#)

[**12.2 HADOOP**](#) 

[12.2.1 Hadoop and MapReduce](#)

[12.2.2 Hadoop EcoSystem](#)

[12.2.3 Hadoop Components](#)

[12.2.4 Hadoop and the Yarn Resource Manager](#)

[12.2.5 PageRank](#)

[12.2.6 INSTALLATION OF HADOOP](#)

[12.2.7 HADOOP VIRTUAL CLUSTER INSTALLATION USING CLOUDMESH](#) 

[**12.3 SPARK**](#) 

[12.3.1 SPARK LECTURES](#) 

[12.3.2 INSTALLATION OF SPARK](#) 

[12.3.3 SPARK STREAMING](#) 

[12.3.4 USER DEFINED FUNCTIONS IN SPARK](#) 

[**12.4 ADVANCED MAPREDUCE TOPICS**](#) 

[12.4.1 AMAZON EMR \(ELASTIC MAP REDUCE\)](#) 

[12.4.2 AMAZON EMR](#) 

[12.4.3 TWISTER](#) 

[12.4.4 TWISTER2 INSTALLATION](#) 

[12.4.5 TWISTER2 EXAMPLES](#) 

[12.4.6 HARP](#) 

[12.4.7 HADOOP RDMA](#)  ?

13 CONTAINERS	⊕
13.1 INTRODUCTION TO CONTAINERS	⊕
13.1.1 Motivation - Microservices	
13.1.2 Motivation - Serverless Computing	
13.1.3 Docker	
13.1.4 Docker and Kubernetes	
13.1.5 Resources	
13.2 DOCKER	⊕
13.2.1 INTRODUCTION TO DOCKER	⊕
13.2.2 RUNNING DOCKER LOCALLY	⊕
13.2.3 DOCKERFILE	⊕
13.2.4 DOCKER HUB	⊕
13.3 DOCKER CLUSTERS ○	⊕
13.3.1 DOCKER SWARM	⊕
13.3.2 DOCKER AND DOCKER SWARM ON FUTURESYSTEMS	⊕
13.3.3 HADOOP WITH DOCKER	⊕
13.3.4 DOCKER PAGERANK	⊕
13.3.5 APACHE SPARK WITH DOCKER	⊕
13.4 KUBERNETES	⊕
13.4.1 INTRODUCTION TO KUBERNETES	⊕
13.4.2 USING KUBERNETES ON FUTURESYSTEMS	⊕
13.5 EXERCISES	⊕
14 SERVERLESS COMPUTING	⊕
14.1 FAAS	⊕
14.1.1 Introduction	
14.1.2 Serverless Computing	
14.1.3 Faas provider	
14.1.4 Resources	
14.1.5 Usage Examples	
14.2 AWS LAMBDA	⊕
14.2.1 AWS Lambda Features	
14.2.2 Understanding Function limitations	
14.2.3 Understanding the free Tier	
14.2.4 Writing your fist Lambda function	
14.2.5 AWS Lambda Usecases	
14.2.6 AWS Lambda Example	
14.3 APACHE OPENWHISK	⊕

14.3.1 OpenWhisk Workflow	•
14.3.2 Setting Up OpenWhisk Locally	•
14.3.3 Hello World in OpenWhisk	•
14.3.4 Creating a custom action	•
14.4 KUBELESS	•
14.4.1 Introduction	•
14.4.2 Programming model	•
14.4.3 System Architecture	•
14.5 MICROSOFT AZURE FUNCTION FA18-516-08	•
14.6 GOOGLE CLOUD FUNCTIONS :SMILEY FA18-516-08	•
14.6.1 Google Cloud Function Example	•
14.7 OPENFAAS :FA18-516-23:	•
14.7.1 OpenFaas Components and Architecture	•
14.7.2 OpenFaas in Action	•
14.8 RIFF ?	•
14.9 FISSION ?	•
14.10 IRONFUNCTION ?	•
14.11 FN ?	•
14.12 GESTALT ?	•
14.13 OPENLAMDA ?	•
14.14 SPRING FUNCTION AS A SERVICE ?	•
15 DEVELOPMENT TOOLS AND SERVICES	•
15.1 REFCARDS	•
15.2 VIRTUAL Box	•
15.2.1 Installation	•
15.2.2 Guest additions	•
15.2.3 Exercises	•
15.3 VAGRANT	•
15.3.1 Installation	•
15.3.2 Usage	•
15.4 PACKER	•
15.4.1 Installation	•
15.4.2 Usage	•
15.5 UBUNTU ON AN USB STICK	•
15.5.1 Ubuntu on an USB stick for macOS via Command Line	•
15.5.2 Ubuntu on an USB stick for macOS via GUI	•
15.5.3 Ubuntu on an USB stick for Windows 10 ?	•

15.5.4 Exercise

15.6 GITHUB

15.6.1 Overview

15.6.2 Upload Key

15.6.3 Fork

15.6.4 Rebase

15.6.5 Remote

15.6.6 Pull Request

15.6.7 Branch

15.6.8 Checkout

15.6.9 Merge

15.6.10 GUI

15.6.11 Windows

15.6.12 Git from the Commandline

15.6.13 Configuration

15.6.14 Upload your public key

15.6.15 Working with a directory that will be provided for you

15.6.16 README.yml and notebook.md

15.6.17 Contributing to the Document

15.6.18 Exercises

15.6.19 Github Issues

15.6.20 GIT PULL REQUEST



15.7 LINUX



15.7.1 History

15.7.2 Shell

15.7.3 Multi-command execution

15.7.4 Keyboard Shortcuts

15.7.5 bashrc and bash_profile

15.7.6 Makefile

15.7.7 chmod

15.7.8 Exercises



15.8 SECURE SHELL



15.8.1 ssh-keygen

15.8.2 ssh-add

15.8.3 SSH Add and Agent

15.8.4 SSH and putty

15.8.5 SSH Port Forwarding O ?

15.8.6 SSH to FutureSystems Resources	⊕
15.8.7 Exercises	⊕
16 DevOps	⊕
16.1 References	⊕
16.2 TRAVIS	⊕
16.2.1 Exercises	⊕
16.2.2 Resources	⊕
16.3 ANSIBLE	⊕
16.3.1 Introduction to Ansible	
16.3.2 Ansible Roles	
16.3.3 Using Variables	
16.3.4 Ansible Galaxy	
16.3.5 A Complete Ansible Galaxy Project	
16.3.6 Exercise	
17 PYTHON	⊕
17.1 INTRODUCTION TO PYTHON	⊕
17.1.1 References	⊕
17.2 PYTHON INSTALLATION	⊕
17.2.1 Managing custom Python installs	
17.2.2 Updating Python Version List	
17.2.3 Updating to a new version of Python with pyenv	
17.2.4 Pyenv in a docker container	
17.2.5 Installation without pyenv	
17.2.6 Anaconda and Miniconda	
17.3 INTERACTIVE PYTHON	⊕
17.3.1 REPL (Read Eval Print Loop)	
17.3.2 Interpreter	
17.3.3 Python 3 Features in Python 2	
17.4 EDITORS	⊕
17.4.1 Pycharm	
17.4.2 Python in 45 minutes	
17.5 LANGUAGE	⊕
17.5.1 Statements and Strings	
17.5.2 Variables	
17.5.3 Data Types	
17.5.4 Module Management	
17.5.5 Date Time in Python	

[17.5.6 Control Statements](#)
[17.5.7 Datatypes](#)
[17.5.8 Functions](#)
[17.5.9 Classes](#)
[17.5.10 Modules](#)
[17.5.11 Lambda Expressions](#)
[17.5.12 Iterators](#)
[17.5.13 Generators](#)
[17.5.14 Non Blocking Threads](#)
[17.5.15 Subprocess](#)
[17.5.16 Queue](#)
[17.5.17 Scheduler](#)
[17.5.18 Python SSL](#)

[17.6 PYTHON MODULES](#)

[17.6.1 Updating Pip](#)
[17.6.2 Using pip to Install Packages](#)
[17.6.3 GUI](#)
[17.6.4 Formatting and Checking Python Code](#)
[17.6.5 Using autopep8](#)
[17.6.6 Writing Python 3 Compatible Code](#)
[17.6.7 Using Python on FutureSystems](#)
[17.6.8 Ecosystem](#)
[17.6.9 Resources](#)
[17.6.10 Exercises](#)

[17.6.11 DATA MANAGEMENT](#)

[17.6.12 PLOTTING WITH MATPLOTLIB](#)

[17.6.13 DocOPTS](#)

[17.6.14 CLOUDMESH COMMAND SHELL](#)

[17.6.15 CMD MODULE](#)

[17.6.16 OPENCV](#)

[17.6.17 SECCHI DISK](#)

[17.6.18 DATA LIBRARIES](#)

[17.7 WORD COUNT WITH PARALLEL PYTHON](#)

[17.7.1 Generating a Document Collection](#)
[17.7.2 Serial Implementation](#)
[17.7.3 Serial Implementation Using map and reduce](#)
[17.7.4 Parallel Implementation](#)

[17.7.5 Benchmarking](#)

[17.7.6 Exercises](#)

[17.7.7 References](#)

[17.8 NUMPY](#)

[17.8.1 Float Range](#)

[17.8.2 Arrays](#)

[17.8.3 Array Operations](#)

[17.8.4 Linear Algebra](#)

[17.8.5 Resources](#)

[17.9 SCIPY](#)

[17.9.1 Introduction](#)

[17.9.2 References](#)

[17.10 SCIKIT-LEARN](#) O

[17.10.1 Installation](#)

[17.10.2 Import](#)

[17.10.3 Create samples](#)

[17.11 Visualize](#)

[17.12 PARALLEL COMPUTING IN PYTHON](#)

[17.12.1 Multi-threading in Python](#)

[17.12.2 Multi-processing in Python](#)

[17.13 DASK](#)

[17.13.1 How Dask Works](#)

[17.13.2 Dask Bag](#)

[17.13.3 Concurrency Features](#)

[17.13.4 Dask Array](#)

[17.13.5 Dask DataFrame](#)

[17.13.6 Dask DataFrame Storage](#)

[17.13.7 Links](#)

[17.14 DASK - RANDOM FOREST FEATURE DETECTION](#)

[17.14.1 Setup](#)

[17.14.2 Dataset](#)

[17.14.3 Detecting Features](#)

[17.14.4 Random Forest](#)

[17.14.5 Acknowledgement](#)

[17.15 FINGERPRINT MATCHING](#) O

[17.15.1 Overview](#)

[17.15.2 Objectives](#)

17.15.3 Prerequisites	
17.15.4 Implementation	
17.15.5 Utility functions	
17.15.6 Dataset	
17.15.7 Data Model	
17.16 Plotting	
17.17 Putting it all Together	
17.18 NIST PEDESTRIAN AND FACE DETECTION	⊕
18 INTRODUCTION TO GO FOR CLOUD COMPUTING	⊕
18.1 Organization of the chapter	
18.2 References	⊕
18.3 INSTALLATION	⊕
18.4 EDITORS SUPPORTING Go	⊕
18.5 Go LANGUAGE	⊕
18.5.1 Concurrency in Go	⊕
18.6 LIBRARIES	⊕
18.7 Go CMD	⊕
18.7.1 CMD	
18.7.2 DocOpts	
18.8 Go REST	⊕
18.8.1 Gorilla	
18.8.2 REST, RESTful	
18.8.3 Router	
18.8.4 Full code	
18.9 OPEN API	⊕
18.9.1 serve specification UI	
18.9.2 validate a specification	
18.9.3 Generate a Go OpenAPI server	
18.9.4 generate a Go OpenAPI client	
18.9.5 generate a spec from the source	
18.9.6 generate a data model	
18.9.7 other editors	
18.9.8 References	
18.10 Go CLOUD	⊕
18.10.1 Golang Openstack Client	
18.10.2 OpenStack from Go	
18.11 Go LINKS	⊕

- [18.11.1 Languages](#)
- [18.11.2 Popular](#)
- [18.11.3 PYPL Popularity of Programming Language](#)
- [18.11.4 Specification to Program](#)
- [18.11.5 Heroku and Go](#)

[19 JULIA O ?](#)



- [19.1 Langugae](#)
- [19.2 Parallel Langauge Constructs](#)
- [19.3 Interfaceing with the System](#)
- [19.4 REST in julia](#)
- [19.5 OPenStack in Julia](#)
- [19.6 AWS in Julia](#)
- [19.7 Azure in Julia](#)
- [19.8 FaaS in Julia](#)

[20 SCALA FOR CLOUD COMPUTING O ?](#)



[21 MESSAGING](#)



[21.1 MQTT](#)



- [21.1.1 Introduction](#)
- [21.1.2 Publish Subscribe Model](#)
- [21.1.3 Secure MQTT Services](#)
- [21.1.4 Integration with Other Services](#)
- [21.1.5 MQTT in Production](#)
- [21.1.6 Installation](#)
- [21.1.7 Server Usecase](#)
- [21.1.8 IoT Use Case with a Raspberry PI](#)
- [21.1.9 Conclusion](#)
- [21.1.10 Exercises](#)

[21.2 GRAPHQL](#)



- [21.2.1 Introduction](#)
- [21.2.2 Prerequisites](#)
- [21.2.3 GraphQL type system and schema](#)
- [21.2.4 GraphQL Query](#)
- [21.2.5 GraphQL in Python](#)
- [21.2.6 Developing your own GraphQL Server](#)
- [21.2.7 Dynamic Queries with GraphQL](#)
- [21.2.8 Advantages of Using GraphQL](#)
- [21.2.9 Disadvantages of Using GraphQL](#)

[21.2.10 Conclusion](#)

[21.2.11 Exercises](#)

[21.3 PYTHON APACHE AVRO](#)

[21.3.1 Download, Unzip and Install](#)

[21.3.2 Defining a schema](#)

[21.3.3 Serializing](#)

[21.3.4 Deserializing](#)

[21.3.5 Resources](#)

[21.4 APACHE FLINK](#) ?

[22 FAQ](#)

[22.1 FAQ: GENERAL](#)

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

[22.1.2 Piazza](#)

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

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

[22.1.5 When contributing to the book my name is not listed properly or not at all](#)

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

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

[22.1.8 Download the epub frequently](#)

[22.1.9 Spelling of filenames in github](#)

[22.1.10 How to open the epub from Github?](#)

[22.1.11 Assignment Summary](#)

[22.1.12 Auto 80 char](#)

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

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

[22.2 FAQ: 516](#)

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

[22.2.2 Opening Epub properly in Linux](#)

[22.2.3 Microsoft Edge for Epub](#)

[22.2.4 Project format: Markdown allowed](#)

[22.2.5 pull request in documents from commandline](#)

[22.2.6 pyenv installed but can not find python](#)

[22.2.7 ssh add/keychain on OSX](#)

22.2.8 check into github early!

22.2.9 I want to start my project now.

23 APPENDIX



23.1 AMAZON WEB SERVICE PRODUCTS

23.1.1 Compute

23.1.2 Storage

23.1.3 Databases

23.1.4 Migration

23.1.5 Networking & Content Delivery

23.1.6 Developer Tools

23.1.7 Management Tools

23.1.8 Media Services

23.1.9 Security, Identity & Compliance

23.1.10 Machine Learning

23.1.11 Analytics

23.1.12 Mobile

23.1.13 AR & VR

23.1.14 Application Integration

23.1.15 Customer Engagement

23.1.16 Business Productivity

23.1.17 Desktop & App Streaming

23.1.18 Internet of Things

23.1.19 Game Development

23.1.20 AWS Marketplace Software

23.1.21 AWS Cost Management

23.1.22 Exercise



23.2 MICROSOFT AZURE AND CLOUD PRODUCTS

23.2.1 AI + Machine Learning

23.2.2 Analytics

23.2.3 Compute

23.2.4 Containers

23.2.5 Databases

23.2.6 Developer Tools

23.2.7 DevOps

23.2.8 Identity

23.2.9 Integration

23.2.10 Internet of Things

- [23.2.11 Management Tools](#)
- [23.2.12 Media](#)
- [23.2.13 Microsoft Azure Stack](#)
- [23.2.14 Migration](#)
- [23.2.15 Mobile](#)
- [23.2.16 Networking](#)
- [23.2.17 Security](#)
- [23.2.18 Storage](#)
- [23.2.19 Web](#)

24 FUTURESYSTEMS

- [24.1 FutureSystems evolved from FutureGrid](#)
- [24.2 Creating Portal Account](#)
- [24.3 SSH Key Generation using ssh-keygen command](#)
- [24.4 Shell Access via SSH](#)
- [24.5 Advanced SSH](#)
- [24.6 SSH Key Generation via putty](#)
- [24.7 FutureSystems Facilities](#)
 - [24.7.1 Bravo](#)
 - [24.7.2 Delta](#)
 - [24.7.3 Echo](#)
 - [24.7.4 Juliet](#)
 - [24.7.5 Romeo](#)
 - [24.7.6 Tango](#)
 - [24.7.7 Tempest](#)
 - [24.7.8 Victor](#)
 - [24.7.9 PI Cluster](#)

25 CHAMELEON CLOUD

- [25.1 CHAMELEON CLOUD SECURITY WARNING](#)
- [25.2 Resources](#)
 - [25.2.1 Outages](#)
 - [25.2.2 Account Creation](#)
 - [25.2.3 Join a Project](#)
 - [25.2.4 Usage Restriction](#)
- [25.3 CHAMELEON CLOUD HARDWARE](#)
 - [25.3.1 Standard Cloud Units](#)
 - [25.3.2 Network](#)
 - [25.3.3 Shared Storage](#)

25.3.4 Heterogeneous Compute Hardware	•
25.3.5 Live updates	•
25.4 CHAMELEON CLOUD CHARGE RATES	•
25.4.1 Service Units	•
25.4.2 Project Allocation Size	•
25.5 GETTING STARTED ON CHAMELEON CLOUD	•
25.5.1 Step 1: Create a Chameleon account	•
25.5.2 Step 2: Create or join a project	•
25.5.3 Step 3: Start using Chameleon	•
25.6 OPENSTACK VIRTUAL MACHINES	•
25.6.1 Web Interface	•
25.6.2 OpenStack REST Interfaces	•
25.6.3 Downloading and uploading data	•
25.7 OPENSTACK COMMAND LINE INTERFACE	•
25.7.1 OpenStack RC File	•
25.7.2 CLI to Manage Virtual Machines	•
25.7.3 Creating SSH keys	○
25.7.4 KeyPair Registration	•
25.7.5 Start a new VM instance	•
25.7.6 Floating IP Address	•
25.7.7 Termination of VM Instance	•
25.8 OPENSTACK HORIZON GRAPHICAL USER INTERFACE	•
25.8.1 Configure resources	•
25.8.2 Interact with resources	•
25.8.3 Use FPGAs	•
25.8.4 Next Step	•
25.9 OPENSTACK HEAT	•
25.9.1 Supporting Complex Appliances	•
25.9.2 Chameleon Appliance Catalog	•
25.9.3 Deployment	•
25.9.4 Heat Template	•
25.9.5 Customizing an existing template	•
25.9.6 Writing a new template	•
25.9.7 Sharing new complex appliances	•
25.9.8 Advanced topics	•
25.10 OPENSTACK BARE METAL	•
25.11 CHAMELEON CLOUD FREQUENTLY ASKED QUESTIONS	•

[25.11.1 Appliances](#)

[25.11.2 Bare Metal Troubleshooting](#)

[25.11.3 OpenStack KVM Troubleshooting](#)

[26. GLOSSARY](#) 

[26.1 VM and Container](#)

[26.2 Network](#)

[26.3 Storage](#)

[27. INCOMING](#) 

[27.1 Box](#) 

[27.1.1 boxpython](#)

[27.1.2 Pybox](#)

[27.2 COMPLIANCE AND GRID COMPUTING](#) 

[27.3 VISUALIZATION](#) 

[27.3.1 Chart Types](#)

[27.3.2 D3 Gallery](#)

[27.3.3 Matplot Gallery](#)

[27.3.4 Bokeh Gallery](#)

[27.3.5 R Gallery](#)

[27.3.6 Gnuplot](#)

[27.3.7 React](#)

[27.3.8 Tablaeu](#)

[27.4 VISUAL STUDIO FOR CLOUD COMPUTING](#) 

[27.5 WINDOWS SUBSYSTEM FOR LINUX](#) 

[28. INDEX](#) 

[REFERENCES](#) 

1 PREFACE



1.1 VERSION



Date: Wed Jan 23 13:36:11 2019 -0500

This document can be downloaded from

- <https://github.com/cloudmesh-community/book/blob/master/vonLaszewski-cloud.epub?raw=true>

1.2 EPUB READERS



This document is distributed in epub format. Every OS will have a suitable epub reader to view the document. Such readers can even be integrated into the browser you use, and when you click on an epub publication in github it could pop up the document in your reader. The way we create the document is through scalable markup, that is you can zoom in and out of a page while the content will be automatically rendered for the selected page size. If you ever see a content that does not fit on a page we recommend you zoom out with your reader to make sure you can see the entire content.

We have made good experiences with the following readers:

- macOS: [Books](#), which is a build in ebook reader
- Windows 10: [Microsoft edge](#), but it must be the newest version, alternatively use [calibre](#)
- Linux: [calibre](#)

1.3 CORRECTIONS



The material collected in this document is managed in

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

In case you see an error or like to make a contribution of your own section or chapter, you can do so in github via pull requests.

The easiest way to fix an error is to read the ePub and click on the cloud  symbol in a heading where you see the error. This will bring you to an editable document. into github). You can directly fix the error in the web browser and create there a pull request. Naturally you need to be signed into github before you can click on the cloud and the redirect will work.

The great thing about doing it this way is that the contributors and authors will be integrated while we inspect with scripts the contributor list the next time we compile the material. Thus even if you corrected a single spelling error, you will be acknowledged.

1.4 CONTRIBUTORS



Contributors are sorted by the first letter of their combined Firstname and Lastname and if not available by their github ID. Please, note that the authors are identified through git logs in addition to some contributors added by hand. The git repository from which this document is derived contains more than the documents included in this document. Thus not everyone in this list may have directly contributed to this document. However if you find someone missing that has contributed (they may not have used this particular git) please let us know. We will add you. The contributors that we are aware of include:

Anand Sriramulu, Ankita Rajendra Alshi, Anthony Duer, Arnav, Averill Cate, Jr, Bertolt Sobolik, Bo Feng, Brad Pope, Dave DeMeulenaere, De'Angelo Rutledge, Eliyah Ben Zayin, Fugang Wang, Geoffrey C. Fox, Gerald Manipon, Gregor von Laszewski, Hyungro Lee, Ian Sims, IzoldalIU, Javier Diaz, Jeevan Reddy Racheppalli, Jonathan Branam, Juliette Zerick, Keli Fine, Mani Kagita,

Miao Jiang, Mihir Shanishchara, Min Chen, Murali Cheruvu, Orly Esteban, Pulasthi Supun, Pulasthi Supun Wickramasinghe, Pulkit Maloo, Qianqian Tang, Ravinder Lambadi, Richa Rastogi, Ritesh Tandon, Saber Sheybani, Sachith Withana, Sandeep Kumar Khandelwal, Silvia Karim, Swarnima H. Sowani, Tim Whitson, Tyler Balson, Vafa Andalibi, Vibhatha Abeykoon, Vineet Barshikar, Yu Luo, ahilgenkamp, aralshi, bfeng, brandonfischer99, btpope, harshadpitkar, himanshu3jul, hrbahramian, isims1, janumudvari, karankotz, qianqian tang, rirasto, shilpasingh21, swsachith, tvangalapat, varunjoshi01, vineetb-gh, xianghang mi

1.5 CREATING THE EPUBS FROM SOURCE



Although you will never likely to create the epub from source, we have included this section for our most advanced contributors and those that update the epub on github.

Please note that you must have at least Pandoc version 2.2.3 installed. You will also need Python version 3.7.1 to run the scripts needed to assamble the document. Earlier versions will not work. You can check the versions with

```
$ pandoc --version  
$ python --version
```

However the easiest way is to use our docker container.

1.5.1 Ubuntu requirements

⚠ this is not yet working as we expect. For now see [Section 1.5.3](#) for an alternative using a virtual machine.

In case you use containers in ubuntu we recommend that you use the docker container to compile the book as discussed in [Section 1.5.5](#).

○ The next is yet untested

In case you like to use the ubuntu system directly, you can download a script that installs the needed software.

You will first have to download the script with

```
$ wget https://raw.githubusercontent.com/cloudmesh-community/book/master/install-ubuntu.sh
```

Than you can run this [script](#) with

```
$ sh install-ubuntu.sh
```

⚠ please note that we have not yet tested this and are looking for feedback and improvements to the `install-ubuntu.sh` script.

Once the software is installed you can scip to [Section 1.5.6](#)

1.5.2 Using Windows 10

We recommend that you use vagrant or docker as described in [Section 1.5.3](#) and [Section 1.5.5](#).

1.5.3 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 loginto the VM with

```
$ vagrant ssh
```

The book folder will be mounted in the VM and you can follow the instructions in [Section 1.5.5](#).

1.5.4 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 1.5.5](#).

1.5.5 Docker

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
```

Now you can skip to [Section 1.5.6](#) 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
```

1.5.6 Creating a book

To create a book, you haе 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/cloud/  
./book/big-data-applications/  
./book/pi  
./book/writing  
./book/222
```

To compile a book go to the directory and make it. Let us assume you like to create the cloud book for 516

```
$ git clone https://github.com/cloudmesh-community/book.git  
$ cd cloud  
$ make new
```

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 new
```

otherwise you can just use

```
$ make
```

The structure of the books is maintained in the yaml file `chapters.yaml`. You can add this chapter to the yaml file, but discuss this first with

Gregor. In case you add a new chapter, you have to say

```
$ make clean  
$ make update  
$ make  
$ make view
```

1.5.7 Publishing the book to github

⚠ This task should only be done by with direct write privileges. and never be part of a pull request. Please discuss with Gregor von Laszewski the details of your update first. Typically Gregor is the one who publishes it.

To publish the book say

```
$ make publish
```

1.5.8 Creating Drafts that are not yet published

Developers of the manual can modify the `Makefile` and locate the variable `DRAFT=` to add additional sections and chapters they work on, but should not yet been distributed with the main publication. Simply add them to the list and say

```
$ make draft  
$ make view
```

to create the draft sections only and view them.

To conveniently call them in a lazy fashion in a terminal you could use the following two aliases.

```
alias m='make; make view'  
alias d='make draft; make view'
```

This allows you to typ `m` for the main volume and `d` for the draft. Please note that all artifacts are written into the dest folder.

1.5.9 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 222 directory.

You have to do the following

```
$ cd 222  
$ make clean  
$ cd ..  
$ cp -r 222 gregor
```

Now edit the file chapters.yaml and copy the section with `book_222=` to `book_gregor=`. Make modifications to the outline as you see fit.

Now you can create the book with

```
$ cd gregor  
$ make update  
$ make new
```

1.6 NOTATION



☁️ :cloud:

If you click on the ☁️ in a heading, you can go directly to the document in github that contains the next content. This is convenient to fix errors or make additions to the content.

\$

Content in bash is marked with verbatim text and a dollar sign

```
$ This is a bash text
```

[1]

References are indicated with a number and are included in the reference chapter [1]

○ :o:

Chapters marked with this emoji are not yet complete or have some issue that we know about. These chapters need to be fixed. If you like to help us fixing this section, please let us know.

🎬 [REST 36:02](#)

Example for a video with the `:clapper:` emoji

🎞 [Slides 10](#)

Example for slides with the `:scroll:` emoji. These slides may or may not include audio.

📝 [Slides 10](#)

Slides without any audio. They may be faster to download.

🎓

A set of learning objectives with the `:mortar_board:` emoji.

✓

A section is release when it is marked with this emoji in the syllabus.

?

Indicates opportunities for contributions.

👉

Indicates sections that are worked on by contributors

😊

Sections marked by the contributor with this emoji when they are ready to be reviewed.



Sections that need modifications are indicated with this emoji.



A warning that we need to look at in more detail.

💡 Notes are indicated with a bulb and are written in italic and surrounded by bars using the `:bulb:` emoji

Figures have a caption and can be referred to in the epub simple with a number. We show such a reference pointer while referring to [Figure 1](#).



Figure 1: Figure example

Figures must be written in the md as

```
![Figure example](images/code.png){#fig:code-example width=1in}
```

You can refer to them with `@fig:code-example`. Please note in order for numbering to work figure references must include the `#fig:` followed by a unique identifier. Please note that identifiers must be really unique and that identifiers such as `#fig:cloud` or similar simple identifiers are a poor choice and will likely not work. To check, please list all lines with an identifier such as

```
$ grep -R "#fig:" chapters
```

and see if your identifier is truly unique.

Other emojis

Other emojis can be found at
<https://gist.github.com/rxaviers/7360908>

⚠ Please note that there is currently a bug when our document is exported to html or to PDF, as emojis are for some reason not properly embedded. Hence to read the document we recommend that you use an ePUB reader.

1.6.1 Hyperlinks in the document

To create hyperlinks in the document other than images, we need to use proper markdown syntax in the source. This is achieved with a reference for example in sections headers. Let us discuss the reference header for this section, e.g. Notation. We have augmented the section header as follows:

```
# Notation {#sec:notation}
```

Now we can use the reference in the text as follows:

```
In #sec:notation we explain ...
```

It will be rendered as: In [Section 1.6](#) we explain ...

1.6.2 Equations

Equations can be written as

```
$$a^2+b^2=c^2$$ {#eq:pythagoras}
```

and used in text:

$$a^2 + b^2 = c^2 \quad (1)$$

It will render as: As we see in [Equation 1](#).

The equation number is optional. Inline equations just use one dollar sign and do not need an equation number:

```
This is the pythagoras theorem: $a^2+b^2=c^2$
```

Whch renders as:

This is the pythagoras theorem: $a^2 + b^2 = c^2$.

1.7 UPDATES



This section lists some updates to the document, spelling and grammare errors are not reported. For a detailed list of updates, please see the Github commits at

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

for a complete list of changes in reverse chronological order

- Jan 23, 2019: Changed the cloud links to view only instead of going directly into the editor. Separated the REST md file into seveal files so we can reorder them and start with OpenAPI instead of restful or eve. Updated the time line in the Syllabus table (see [Section 3.4](#))
- Jan 21, 2019: Added gitissues separated by labels.
- Jan 18, 2019: A section on how to do equation in the book was added (see [Section 1.6.2](#)). Updated the Datacenter Section.
- Jan 16, 2019: Added the example about defining a REST service based on an OpenAPI yaml file (see [Section 9.4](#)).
- Jan 15, 2019: Slight improvements in the preface: Added this Update section; Added explanation ho to do references in markdown in Section Notation; Updated the github issues in the preface; remove the obsolete abbreviation section for the inclusion of emojis which is now handled via a script
- Jan 15, 2019: Slight improvements in the Qucik Start: Removed the redundant project definition; Added the

Technology ePub link; simplified the Project description; Clarified the use of the Workbreakdown section and added todo items to the Workbreakdown section requirements for better interaction in discussions with TAs throughout the semester.

- Jan 15, 2019: experimented On OSX on the install of pandoc via cabal and installed pandoc-citeproc via cabal. This is documented at

<https://github.com/jgm/pandoc-citeproc>

This breaks instalation instructions for Linux/ubuntu. For ubuntu we have created an experimental untested install script if containers can not be used.

1.8 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. Time sensitive items such as assignment due dates are listed in another table.

○ add links to sections

Here we list some of the activities by week.

1.8.1 Week 1: Activities Jan 10 - 17

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 REAME.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 3.7.1. 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 is solely delegated 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.

1.8.2 Week 2: Jan 17 - 25

1.8.2.1 Development machine

If you have not yet set up a computer with python 3.7.1 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

1.8.2.2 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

1.8.2.2.1 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.

1.8.2.2.2 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.

1.8.2.2.3 Datacenter Exercises

Some of you asked what Exercises they can do for this week to do some work beyond reading the section.

We suggest you do

- E.Carbon.2 (see [Section 6.5.2](#))
- E.Energy.1 (see [Section 6.12](#))
- E.Energy.2 (see [Section 6.12](#))

1.8.2.3 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.

1.8.2.4 Python till rest of the semester

Python 3.7.1: Set up a computer on which you can execute python 3.7.1. Some did not complete that task yet (see [Section 17.2](#), [Section 17.1](#), [Section 17.5](#)).

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 3.7.1 when executing the python programs do it in a terminal not from within pycharm (see

[Section 17.4](#)).

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 extreamly helpful.

1.8.3 Week 3: Jan 25 - Feb 1

1.8.3.1 Cloud Architectures

This week we will focus on archie=tectural 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 asnwer and add it to the handbook at the spots marked with a red circle.

Read the sections:

- Architectures (see [Section 7](#))
- NIST Big Data Reference Architecture (see [Section 8](#))

Students and TA's please complete the red cicles, e.g. mostly bibtex refernces.

1.8.3.2 REST services

Read the REST Service section. THi section includes some parts that are not that relevant for this class and we llike you to focus on in [Section 9](#) on the sections

- Overview
- OpenAPI REST Services with Swagger (see [Section 9.2](#))
- OpenAPI Specification (see [Section 9.3](#))
- OpenAPI REST Service via Introspection (see sec:openapi-

introspection)

You do not in thi 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.

1.8.4 Week 4: Feb 1 - Feb 8

1.8.4.1 Github as REST service

Read:

- Github REST Services ([Section 9.11](#))

Recommended:

- Do E.github.issues.6 as this is directly relevant for your projects and can be generalized to other REST services.

Optional:

- Do excersise E.github.issues.5:

⚠ The week 4 is under construction and additional items will be added

1.8.4.2 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.

2 GITHUB Book ISSUES



In this section we find the issues that we file in github to coordinate the improvement of the document.

2.1 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.

2.1.1 Internal Issues

.

N	#	Title	Assignee	Labels
10	290	bibtex issues	bfeng	internal
11	289	add to datacenter section	tbalson	internal
12	288	HIGHEST PRIORITY Missing BibTex in figure captions	bfeng	internal
16	276	Add weekly activities into the handbook	bfeng	internal
21	267	figure tags with bibtex ref	bfeng	internal
22	266	Internal: Github video use	tbalson	internal
23	265	Internal: Check 222 book order	tbalson	e222 internal
34	254	Fingerprint matching	pulasthi	internal section
35	253	Ubuntu on an USB stick for Windows 10	None	internal section
47	239	Internal: New trends	pulasthi	internal
48	238	internal: readme on anchoring links	tbalson	internal
49	237	internal: piazza scribe for 222	tbalson	internal
51	235	Internal: Open sections and chapters	pulasthi	internal
55	231	above, below	bfeng	internal
56	227	internal: Virtual machine .md file is missing	tbalson	bug internal

58	211	some refs missing	pulasthi	bug internal
59	190	internal: Hadoop 3.1.1	bfeng	bug internal
61	186	internal: figure references {#fig:label}	bfeng	internal
63	180	E534 Cloud Section lecture videos merged into E-books	None	assigned internal
64	173	internal: biber installation	bfeng	internal
		internal: Upload lecture videos from Prof Fox's Google Drive to Youtube	laszewsk	internal
69	144	internal: bigdata missing improvements	laszewsk	internal
71	134	internal: bibtex errors in technologies	bfeng	internal
72	126	bibtex: 523	bfeng	internal
		Check and redo Docker Hadoop 3.0.1 and Docker for Hadoop section.	bfeng	internal
74	110	replace images with text	bfeng	internal
75	77	add mapreduce to syllabus	laszewsk	assigned internal
76	76	openapi demo	laszewsk	assigned internal
77	68	internal: bib: create bibtex entries for go-intro.md	bfeng	internal
82	47	Internal: Hot cold isles in data center	pulasthi	assigned internal

2.1.2 Open Sections

N	#	Title	Assignee	Labels
1	302	MongoDB	pulasthi	chapter open section
5	298	Pi: docker and grafana and influxdb	None	open section
25	263	Open: Singularity	pulasthi	chapter open section
26	262	Open: Windows subsystems for Linux	None	open section
27	261	Low Priority: Visual Studio for Cloud Computing (PyCharm is better)	None	e222 open section
30	258	Open: Apache Flink	None	chapter open section
36	252	Packer	None	open section
37	251	OpenLambda	None	open section
38	250	Fn Project	None	open section
39	249	IronFunctions	None	open section

40	248	Fission	None	open section
41	247	Riff	None	open section
42	246	OpenFaaS	None	open section
43	245	Hadoop virtual cluster installation using cloudmesh	None	open section
44	244	Google IaaS Cloud Services	None	open section
45	243	Artificial intelligence service with REST	None	open section
46	242	Scikit-Learn	None	open section
53	233	Section: AWS DocumentDB	None	open section
54	232	Section: Port forwarding	ameet20	open section
57	223	Open: Section CircleCi	None	open section

2.1.3 Open Chapters

N	#	Title	Assignee	Labels
1	302	MongoDB	pulasthi	chapter open section
25	263	Open: Singularity	pulasthi	chapter open

25	263	Open: Singularity	pulasthi	section
28	260	Open: Cloud foundry	None	chapter open
30	258	Open: Apache Flink	None	chapter open section
31	257	Open: Apache Avro (Python)	None	chapter open
33	255	Julia	None	chapter open
50	236	Open: EMR section(s)	None	chapter open

2.1.4 Assigned Sections

N	#	Title	Assignee	Labels
8	292	Section: Infrastructure as Code - Terraform	joshish-iu	assigned section
13	286	Section: Data Services: AWS RedShift	joshish-iu	assigned section
20	268	Open: Google Drive Python interface	g14uok	assigned chapter section

2.1.5 Assigned Chapters

N	#	Title	Assignee	Labels
19	272	Assigned: Puppet	tandon-ritz	assigned

20	268	Open: Google Drive Python interface	g14uok	assigned chapter section
29	259	Open: Box	kelifine	assigned chapter
32	256	Assigned: Chapter & Sections: Scala for cloud computing	hrbahramian	assigned chapter

2.1.6 Open Projects

N	#	Title	Assignee	Labels
52	234	Project: Virtual Cloud Directory	None	open project

2.1.7 Assigned Projects

N	\#	Title	Assignee	Labels
---	----	-------	----------	--------

3 QUICK START



This quick start will help you to get started in this class and identify one out of three mechanisms that you can chose to take this class.

3.1 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.

3.2 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 three 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. The sections that are released are marked with . Note that the book may include sections that are not listed in the syllabus. You do not have to read such sections.
- 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. On weekly basis we may add a new section. Such sections are marked for one week with the icon after about three weeks time all icons will be removed. The icon is there to help you identifying sections that may have been added to section that logically fit better in an earlier section, than linearly at the end

- Way 3: Weekly guided progress. Other students may prefer a more guided approach to the class. Thus we will provide a summary of what you can do on a weekly activity to stay up to date. We will add to the weekly schedule once new items become available. We also augment them with experiments/exercises that are not graded or reviewed, but that you could discuss with the TAs. More details are posted in the syllabus of this class part of this document.

3.3 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.

3.3.1 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.

3.3.2 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 **O**. 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/vonl/cloud-technologies.epub?raw=true>

3.3.3 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-community/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.

3.3.4 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.

3.3.4.1 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 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 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

3.3.5 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.

3.3.6 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 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
```

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.

3.3.7 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, 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).

3.4 Course Syllabus Tables



Legend markings

- Class released ✓ |
- Class under development o

o Warning the dates may change not yet correct.

- Semester Start for this class 01/07
- Semester end for this class 05/03, all assignments are due 2 weeks before semester end.

3.4.1 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.

Dates	Unit	Title	Description
✓ 01/07	1	Introduction	Gregor von Laszewski
✓ 01/07			Class summary
✓ 01/07			Definition of Cloud Computing
✓ 01/07	2	Tools	Tools and Services
✓ 01/07			- Virtual Box
✓ 01/07			- Vagrant
✓ 01/07			- Github
✓ 01/07			- Linux
✓ 01/14	3	Python	Python
✓ 01/14			- Introduction
✓ 01/14			- Installation
✓ 01/14			- Interactive Python
✓ 01/14			- Editors
✓ 01/14			- Basic Language Features
✓ 01/14			- Modules
✓ 01/14			- Data Management
✓ 01/14			- Matplotlib
✓ 01/14			- Cloudmesh Commandshell

✓ 01/14

CMD5

✓ 01/14	- OpenCV
✓ 01/14	- Secchi Disk
✓ 01/18	Data Center
✓ 01/21 4 Architectures	- NIST Big Data Reference Architecture
✓ 01/21	- Cloud Architectures
✓ 01/25	REST
✓ 01/25	- OpenAPI and Swagger
✓ 01/25	- OpenAPI Specification
✓ 01/25	- OpenAPI Service
✓ 01/25	- Github as Rest Service
✓ 02/01 5 Virtualization	Virtualization, Qemu, KVM, Virtual machines
✓ 02/01 5 Virtualization I	- Qemu
✓ 02/01 6 Infrastructure	Infrastructure as a Service
✓ 02/01	- Azure
✓ 02/01	- AWS
✓ 02/01	- OpenStack
✓ 02/04 Chameleon Cloud	- Chameleon Cloud
✓ 02/04	- Resources
✓ 02/04	- Hardware
✓ 02/04	- Charge
✓ 02/04	- Quick start

✓	02/04		- CLI
✓	02/04		- Horizon
✓	02/04		- Heat
✓	02/04		- Baremetal
✓	02/04		- FAQ
✓	02/11	8	Programming Python for Cloud Computing,
✓	02/11		- Libcloud
o	02/11		Virtualization II Containers, Docker, Kubernetes
o	02/18	9	Map/Reduce Map/Reduce, Hadoop, Spark
✓	02/25	10	Messaging Messaging
✓	02/25	11	Messaging - MQTT
✓	03/04		- Graphql
o	04/01		Go Introduction
o	04/01		- Go Links
o	04/01		- Go Install
✓	04/01		- Go Editors
o	04/01		- Go Language
o	04/01		- Go Libraries
o	04/01		- Go cmd
o	04/01		- Go Cloud
o	04/01		- Go REST
o	04/01		- Go for the Cloud
o	04/15	13	Edge Computing Edge Computing and the Cloud

3.4.2 Proposed Assignments Timeline

Dates	Unit	Title	Description
✓ 01/22-03/17	A0	Survey	Fill out the Survey before Friday in the first week
✓ 01/22-03/17	A0	Bio	Post your formal Bio into Piazza
✓ 01/22-03/17	A1	Sections	Contribute significant sections. Do not develop redundant or duplicated content.
✓ 01/22-03/17	A2	Chapter	Contribute a significant chapter that may use your section to the class documentation. Do not develop redundant or duplicated content.
✓ 01/22-03/17	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 break.
✓ 01/22-04/21	A3	Project Type A	Build a cloud cluster out of Raspberry Pis
✓ 01/22-04/21		Project Type B	Build a Significant OpenAPI REST Service
✓ 01/22-04/21		Project Type C	Build an Edge Service Interfacing with a Cloud
✓ 01/22-04/21		Project Type D	Contribute to the new Cloudmesh code
✓ 01/22-04/21		Project Type E	Your own Project Type A, B, C, D [upon approval)

You have to do only one project type.

Students need only to do one project. The project is conducted thought the entire semester.

- Dates may change as the semester evolves
- 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	

✓ anytime	Writing a Project Report	How to write a high quality Project report following our template
✓ anytime	Bibliography Management	How to easily manage bibliographies for your Project Report

3.4.3 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 disproportionately 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. E.g. a project containing for example 3 members should contribute the work of 3 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.



4.1 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 2](#) 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.

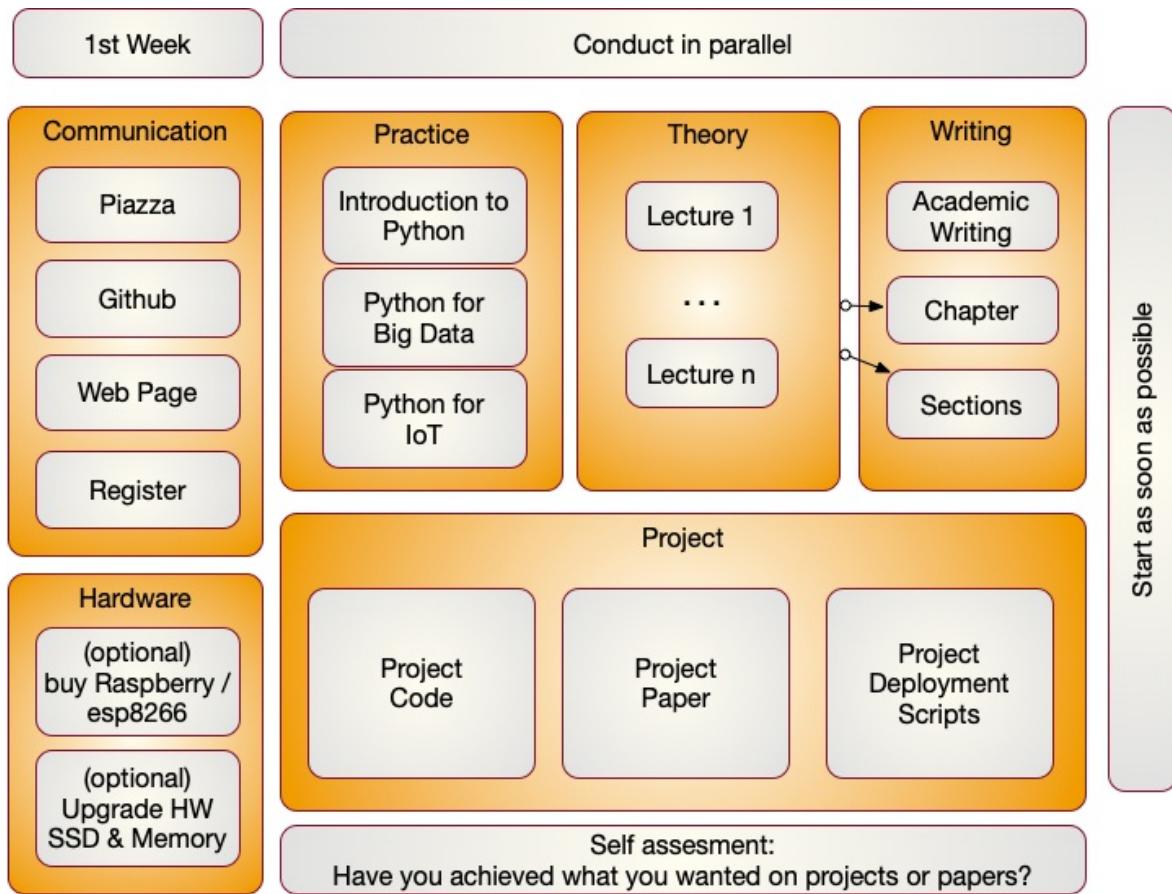


Figure 2: Components of the Class e516, e534 and e616

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.

4.1.1 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.

4.1.2 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.

4.1.3 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.

4.1.4 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 off main memory which gives you enough space to run containers, virtual machines and naturally the main operating system. We fond 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, your could also get yourself a UBS stick or external SSD drive and place ubuntu on it. You could than 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.

4.1.4.1 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.

4.1.4.2 Fun

I hope you have fun and are able to integrate in the projects your

own thoughts and interests.

We have quotes from 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. They 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.

4.1.4.3 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.

4.1.4.4 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.

4.1.5 Parallel Tracks

In this class we have three parallel tracks.

4.1.5.1 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 than 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.

4.1.5.2 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.

4.1.5.3 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.

4.1.5.4 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.

4.1.6 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.

4.2 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 \(14:13\)](#)

4.3 E516: ENGINEERING CLOUD COMPUTING



⚠ Students considering to take this calls in Spring 2019 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 teh 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/spring2019/e516spring19/resources>

- 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.

4.3.1 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.

4.3.2 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

4.3.3 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.

4.4 SYLLABUS

For the syllabus table please see the the syllabus table

4.4.1 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 major 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 Dec 7th.

4.4.1.1 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.

4.5 COURSE POLICIES



We describe briefly some class policies.

4.5.1 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>

4.5.2 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.

4.5.3 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.

4.5.3.1 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

4.5.4 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.

4.5.5 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.

4.5.6 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.

4.5.7 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.

4.5.8 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.

4.5.9 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.

4.5.10 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.

4.5.11 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.

4.5.12 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.

4.5.13 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 planning. 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 months to complete it. We will need 2 months 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>

4.5.13.1 Exercises

E.Policy.1

Take the Plagiarism test, See the Scientific Writing I
epub for more details.

4.6 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](#)

4.6.1 Introduction

4.6.1.1 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

4.6.2 Class communication

We are using piazza and github for class communication. 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.

4.6.2.1 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
 - DocOpt
 - 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
 - DocOpt 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 pieces 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.

4.6.3 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 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 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.

4.6.4 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.

4.7 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 student's previous work without significant improvements. All material listed here is available online, including all source code.

4.7.1 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/vonl-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/te>

Ex - Title Of Summary ✋ fa18-xxx-xx

4.7.2 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 below

- 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->

<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.

4.7.3 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>

5 DEFINITION OF CLOUD COMPUTING



🎓 Learning Objectives

- Compare different definitions of cloud computing.
- Review the History of cloud computing.
- Identify trends.
- The current hot job is data engineer which is sought after more than data scientists (a new trend). You have chosen the right course 😊
- Be TALLL to be successful in cloud computing.

We are looking at some of the definitions of cloud computing and trends in the following presentation.

- 🎬 [Definition of Cloud Computing](#)

This includes the following

5.1 DEFINING THE TERM CLOUD COMPUTING

In this presentation we review three definitions of cloud computing. This includes the definitions by

- NIST
- Wikipedia
- Gartner

5.2 HISTORY AND TRENDS

We review some of the historical aspects that lead to cloud computing and especially look into more recent trends. These trends motivate that we need to look at enhancements to the traditional Service Model that include Infrastructure-, Platform- and Software- as

a Service. These enhancements especially are targeting Function-, and Container as a Service.

5.3 JOB AS A DATA ENGINEER

We look at some job related trends that especially focus on the newest hot job description called **Data Engineer**. It is motivated that current job offerings as data engineer is 13% versus 1% for data scientists. As this class is targeted towards bringing the engineering component towards the data scientists, computer scientists, and application developer, This class is ideally suited for increasing your marketability.

5.4 YOU MUST BE THAT TALLL

We close this class with Gregor's TALLL principle to succeed in Cloud Computing:

You must be that TALLL to survive in Cloud Computing and Big Data

This principle includes the following characteristics

Trend Awareness (TA)

We need to be aware not only what is currently a trend, but what will be future trends

Longevity Planning (L)

We need to be able to reproduce our services and results (e.g. can we reproduce them still in six month).

Leap Detection (L)

We need to be able to deal with technology Leaps

Learning Willingness (L)

We need to constantly learn to keep up as technology changes every 6 month

Naturally this principal can be applied to other disciplines.



❖ Learning Objectives

- What is a data center.
- What are import metrics.
- What is the difference between a Cloud data center and a traditional datacenter.
- What are examples of Cloud data centers.

6.1 MOTIVATION: DATA

Before we go into the details of a data center we like to motivate why we need them. Here we start with looking at the amount of data that recently got created and provide one of many motivational aspects. Not all data will or should be stored in data centers. However a significant amount of data will be in such centers.

6.1.1 How much data?

One of the issues we have is to actually comprehending how much data is actually created. As humans the total sum created over a year is hard to imagine and put into a perspective that can easily be understood. In order to visualize the data produced we find often Graphics about how much i created in one minute instead. Such depictions include samples of data created as a part of popular cloud services or the internet in general.

One such popular depiction is Data Never Sleeps (see [Figure 3](#)). It has been produced a number of times over the years and is now at version 6.0 released in 2017. If you identify a newer version, please let us know. It is worth while to study this image in detail and identify some of the data that you can relate to of service you use. It is also a possible indication to study other services that are mentioned. A

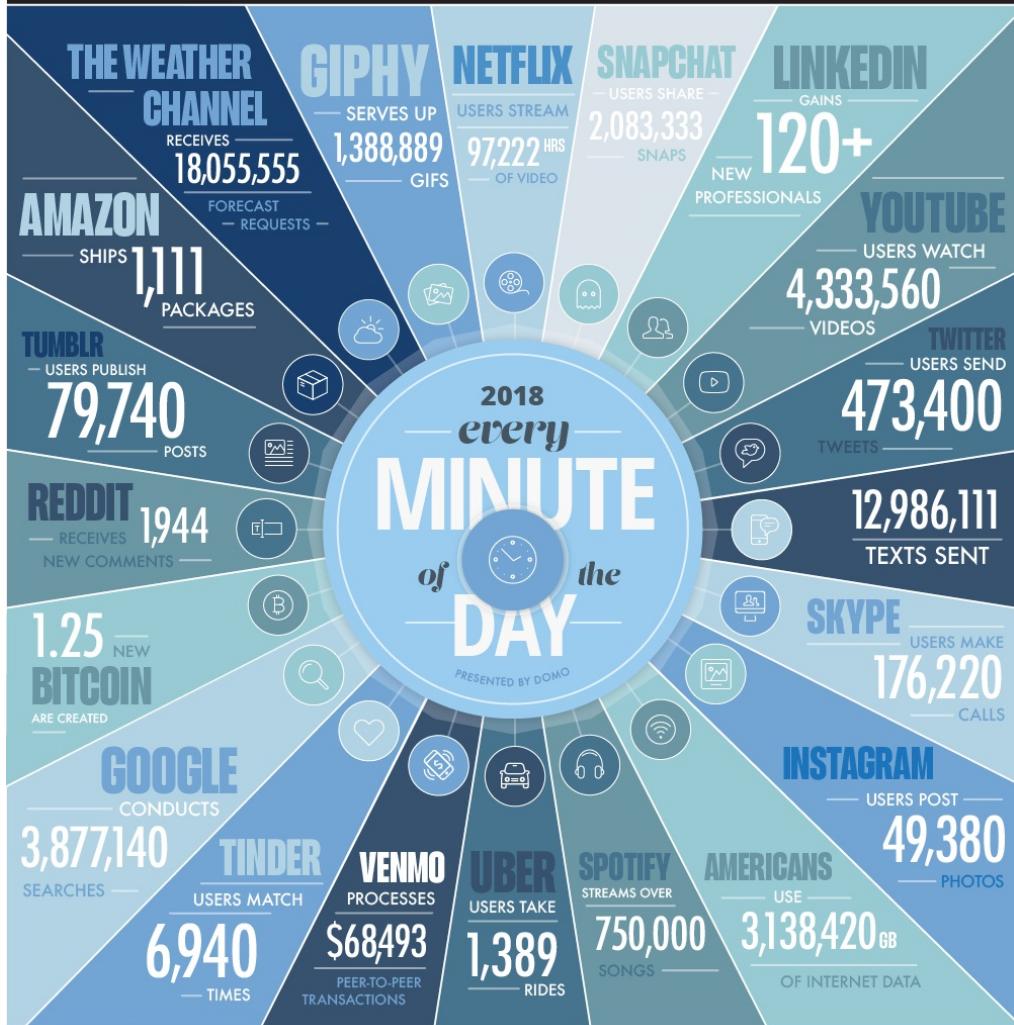
staggering 3.8Mil google searches are executed every minute. Surprisingly the weather channel receives over 18Mil forecast requests which is even higher than the 12Mil text messages send every minute. Youtube certainly serving a significant number of users by 4.3Mil videos watched every minute. Naturally the numbers are averages over time.

DOMO

DATA NEVER SLEEPS 6.0

How much data is generated *every minute*?

There's no way around it: big data just keeps getting bigger. The numbers are staggering, but they're not slowing down. By 2020, it's estimated that for every person on earth, 1.7 MB of data will be created every second. In our 6th edition of Data Never Sleeps, we once again take a look at how much data is being created all around us every single minute of the day—and we have a feeling things are just getting started.



The world's internet population is growing significantly year-over-year. In 2017, internet usage reached 47% of the world's population and now represents 3.8 billion people.



The ability to make data-driven decisions is crucial to any business. With each click, swipe, share, and like, a world of valuable information is created. Domo puts the power to make those decisions right into the palm of your hand by connecting your data and your people at any moment, on any device, so they can make the kind of decisions that make an impact.

Learn more at domo.com

SOURCES: STATISTA, LINKEDIN, INTERNET LIVE STATS, EXPANDED RAMBLINGS, SLASH FILM, RIAA, BUSINESS OF APPS, INTERNATIONAL TELECOMMUNICATIONS UNION, INTERNATIONAL DATA CORPORATION



Figure 3: Data Never Sleeps [2]

A different source publishes what is happening on the internet in a minute, but we have been able to locate a version from 2018 (see [Figure 4](#)). While some data seems the same, others are slightly different. For example this graph has a lower count for Google searches, while the number of text messages send is significantly higher in contrast to [Figure 3](#).

2018 This Is What Happens In An Internet Minute



Figure 4: Internet Minute 2018 [3]

While reviewing the image from last year from the same author, we find not only increases, but also declines. Looking at facebook showcases a loss of 73000 logins per minute. This loss is substantial. We can see that facebook services are replaced by other services that are more popular with the younger generation who tend to pick up new services quickly (see [Figure 5](#)).

2017 This Is What Happens In An Internet Minute



2018 This Is What Happens In An Internet Minute



Figure 5: Internet Minute 2017-2018 [3]

It is also interesting to compare such trends over a longer period of time (see [Figure 6](#), [Figure 7](#)). An example is provided by looking at Google searches

- <http://www.internetlivestats.com/google-search-statistics/>.

and visualized in [Figure 6](#).

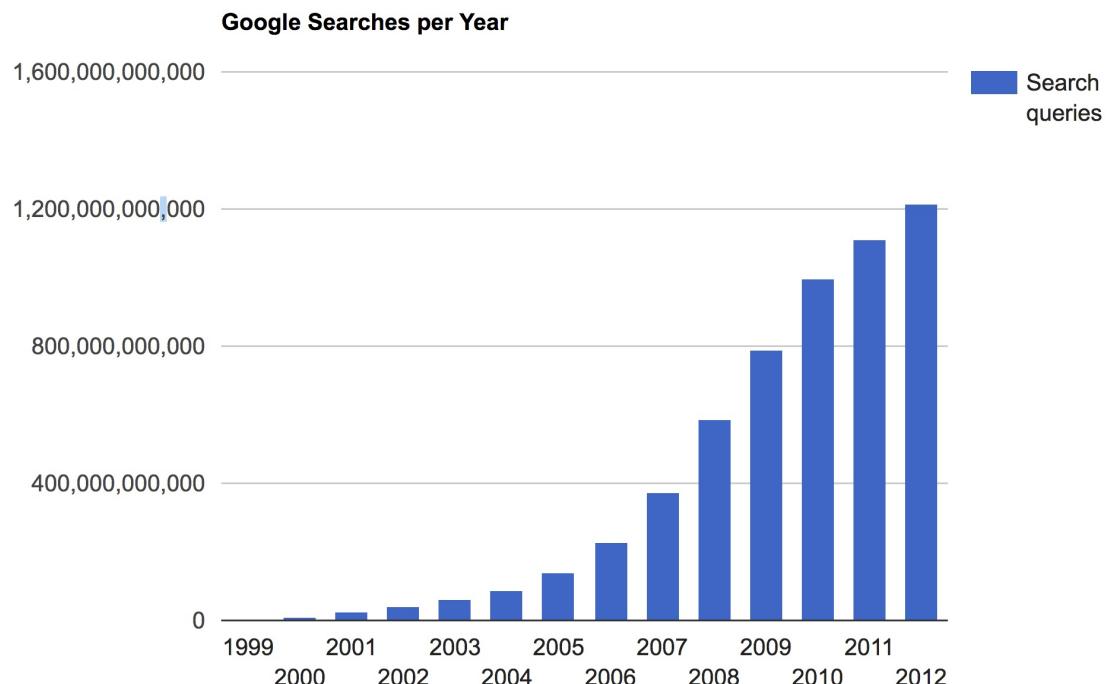


Figure 6: Google searches over time

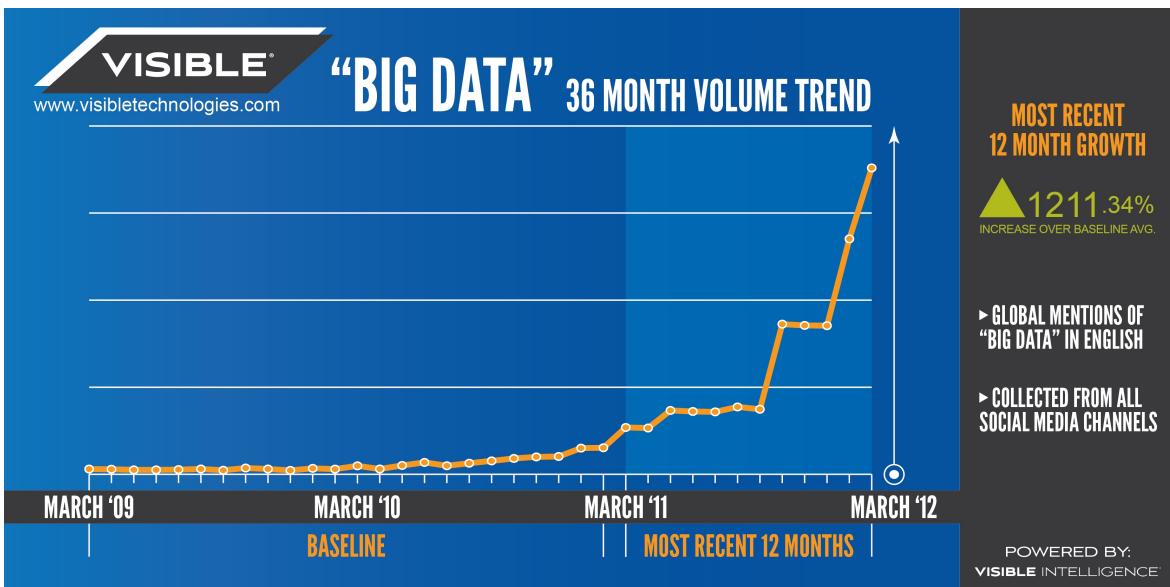


Figure 7: Big data trend. 2012 [4]

When looking at the trends, many predict an exponential growth in data. This trend is continuing.

6.2 CLOUD DATA CENTERS

A data center is a facility that hosts the information technology related to servers and data serving a large number of customers. Data centers evolved from the need to originally have large rooms as the original computers filled in the early days of the computer revolution filled rooms. Once multiple computers were added to such facilities super computer centers created for research purposes. With the introduction of the internet and offering services such as Web hosting large business oriented server rooms were created. The need for increased facilities was even accelerated by the development of virtualization and servers being rented to customers in shared facilities. As the need of web hosting still is important but has been taken over by cloud data centers, the terms internet data center, and cloud data center are no longer used to distinguish it. Instead we use today just the term data center. There may be still an important difference between research data centers offered in academia and industry that focus on providing computationally potent clusters focus on numerical computation. Such data centers are typically

centered around the governance around a smaller number of users that are either part of an organization or a virtual organization. However, we see that even in the research community data centers not only host supercomputers, but also Web server infrastructure and these days even private clouds that support the organizational users. In case of the latter we speak about supporting the long tail about science.

The later is driven by the 80%-20% rule. E.g. 20% of the users use 80% of the compute power. This means that the top 20% of scientists are served by the leadership class super computers in the nation, while the rest are either served by other servers, cloud offerings through research and public clouds.

6.3 DATA CENTER INFRASTRUCTURE

Due to the data and the server needs in th cloud and in research such data centers may look very different. Some focus on large scale computational resources, some on commodity hardware offered to the community. The size of them is also very different. While a supercomputing center as part of a University was one of the largest such data centers two decades ago, they dwarf the centers now deployed by industry to server the long tail of customers.

In general a data center will have the following components.

- Facility: the entire data center will be hosted in a building. The building may have specific requirements related to security, environmental concerns, or even the integration into the local community with for example providing heat to surrounding residences.
- Support infrastructure: This building will include a significant number of support infrastructure that addresses for example continuous power supply, air conditioning, and security For this reason you find in such centers

- Uninterruptible Power Sources (UPS)
 - Environmental Control Units
 - Physical Security Systems
- Information Technology Equipment: Naturally the facility will host the IT equipment including the following:
 - Servers
 - Network Services
 - Disks
 - Data Backup Services
- Operations staff: The facility will need to be staffed with the various groups that support such data centers. It includes
 - IT Staff
 - Security and Facility Staff
 - Support Infrastructure Staff

With regards to the number of people serving such a facility it is obvious that through automation is quite low. According to [5] proper data center staffing is a key to a reliable operation (see [Figure 8](#)).

According to [Figure 8](#) operational sustainability contains three elements of operational sustainability, namely management and operations, building characteristics, and site location [5].

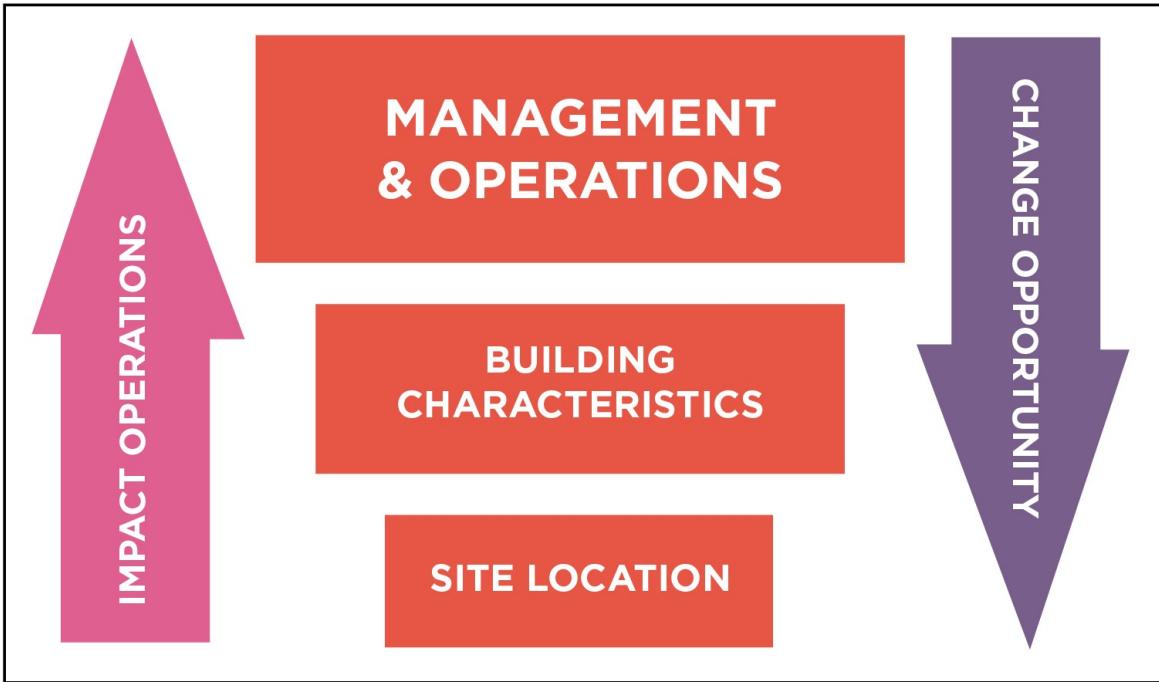


Figure 8: Datacenter Staff Impact [5]

Another interesting observation is the root cause of incidents in a data center. Everyone has probably experienced some outage, so it is important to identify where they come from in order to prevent them. As we see in [Figure 9](#) not every error is caused by an operational issue. External, installation, design and manufacturer issues are together the largest issue for datacenter incidents (see [Figure 9](#)). Figure Outage. According to the Uptime Institute Abnormal Incident Reports (AIRs) database, the root cause of 39% of data center incidents falls into the operational area [5].

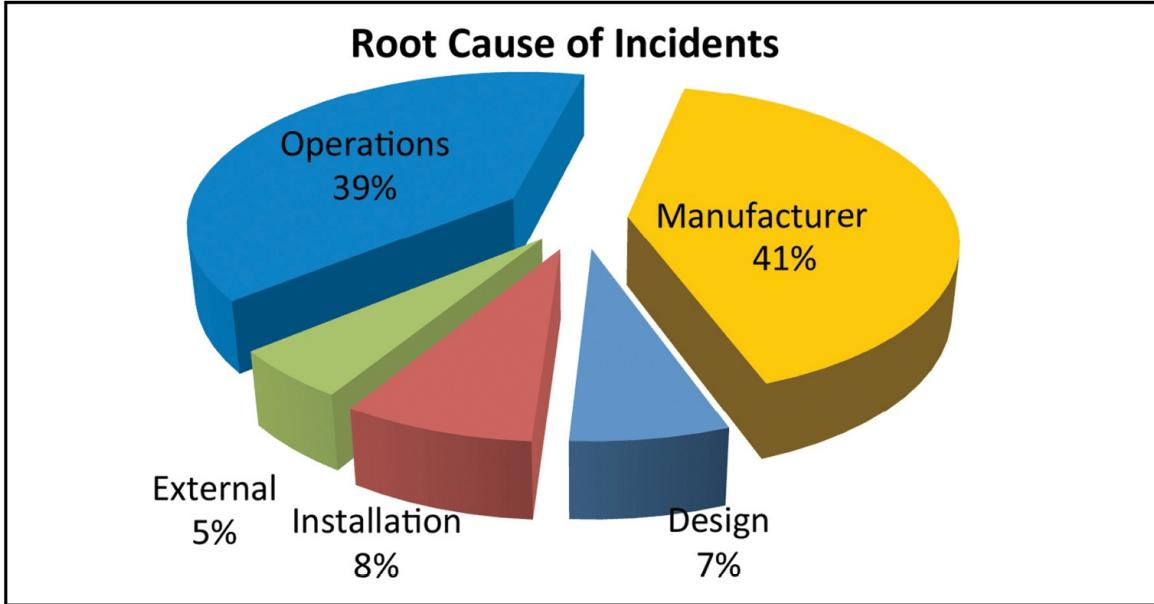


Figure 9: Datacenter outage [5]

6.4 DATA CENTER CHARACTERISTICS

Next we identify a number of characteristics when looking at different data centers.

Variation in Size: Data centers range in size from small edge facilities to megascale or hyperscale filling large ware houses.

Variation in cost per server: Although many data centers standardize their components, specialized services may be afford not on a 1K server, but on a 50K server.

Variation in Infrastructure: Servers in centers serve a variation of needs and motivate different infrastructure: Use cases, Web Server, E-mail, Machine Learning, Pleasantly Parallel problem, traditional super computing jobs.

Energy Cost: Data centers use a lot of energy. The energy cost varies per region. A motivation to reduce energy use and cost is also been trended by environmental awareness, not only by the operators, but by the community in which such centers operate.

Reliability: Although through operation I efforts the data center can be made more reliable, failure still can happen. Examples are

- <https://www.zdnet.com/article/microsoft-south-central-u-s-datacenter-outage-takes-down-a-number-of-cloud-services/>
- ○ find more examples

Hence Data Center IaaS advantages include

- Reduced operational cost
- Increased reliability
- Increased scalability
- Increased flexibility
- Increased support
- Rapid deployment
- Decrease management: Outsourcing expertise that is not related to core business

Datacenter disadvantages include

- Loss of control of the HW
- Loss of control of the data
- Model is preferring many users
- Software to control infrastructure is not accessible
- Variations in performance due to sharing
- Integration requires effort beyond login
- Failures can have a humongous impact

6.5 DATA CENTER METRICS

One of the important factors for a smooth operation but also for a smooth offering of services is to employ metrics that will be able to provide significant impacting the operations. Having metrics allows the staff to monitor and adapt to dynamic situations but also to plan operations.

6.5.1 Data Center Carbon Footprint

Scientists world wide have identified a link between carbon emission and global warming. As the energy consumption of a data center is substantial, it is prudent to estimate the overall carbon emission. Schneider Electric (formerly APC) has provided a report on how to estimate the Carbon footprint of a data center [6]. Although this report is already a bit older, it provides still valuable information. It defines key terms such as

Carbon dioxide emissions coefficient (carbon footprint):

With the increasing demand of data, bandwidth and high performance systems, there is substantial amount of power consumption. This leads to high amount of greenhouses gases emission into the atmosphere, released due to any kind of basic activities like driving a vehicle or running a power plant.

"The measurement includes power generation plus transmission and distribution losses incurred during delivery of the electricity to its point of use."

Data centers in total used 91 billion kilowatt-hours (kWh) of electrical energy in 2013, and they will use 139 billion kWh by 2020. Currently, data centers consume up to 3 percent of all global electricity production while producing 200 million metric tons of carbon dioxide. Since world is moving towards cloud, causing more and more data center capacity leading more to power consumption.

Peaker plant:

Peaking power plants, also known as peaker plants, and occasionally just peakers, are power plants that generally run only when there is a high demand, known as peak demand, for electricity. Because they supply power only occasionally, the power supplied commands a much higher price per kilowatt hour than base load power. Peak load power plants are dispatched in combination with base load power plants, which supply a dependable and consistent amount of electricity, to meet the minimum demand. These plants are generally coal-fired which

causes a huge amount of CO₂ emissions. A peaker plant may operate many hours a day, or it may operate only a few hours per year, depending on the condition of the region's electrical grid. Because of the cost of building an efficient power plant, if a peaker plant is only going to be run for a short or highly variable time, it does not make economic sense to make it as efficient as a base load power plant. In addition, the equipment and fuels used in base load plants are often unsuitable for use in peaker plants because the fluctuating conditions would severely strain the equipment. For these reasons, nuclear, geothermal, waste-to-energy, coal and biomass are rarely, if ever, operated as peaker plants.

Avoided emissions:

Emissions avoidance is the most effective carbon management strategy over a multi-decadal timescale to achieve atmospheric CO₂ stabilization and a subsequent decline. This prevents, in the first place, stable underground carbon deposits from entering either the atmosphere or less stable carbon pools on land and in the oceans.

Carbon offsets based on energy efficiency rely on technical efficiencies to reduce energy consumption and therefore reduce CO₂ emissions. Such improvements are often achieved by introducing more energy efficient lighting, cooking, heating and cooling systems. These are real emission reduction strategies and have created valid offset projects.

This type of carbon offset provides perhaps the simplest options that will ease the adoption of low carbon practice. When these practices become generally accepted (or compulsory), they will no longer qualify as offsets and further efficiencies will need to be promoted.

CO₂ (carbon dioxide, or carbon)

Carbon dioxide is the main cause of the greenhouse effect, it is emitted in huge amount into our atmosphere with a life cycle of

almost 100 years. Data centers emit during the manufacturing process of all the components that populate a data center (servers, UPS, building shell, cooling, etc.) and during operation of data centers (in terms of electricity consumed), the maintenance of the data centers (i.e. replacement of consumables like batteries, capacitors, etc.), and the disposal of the components of the data centers at the end of the lifecycle. Until now, power plants have been allowed to dump unlimited amounts of carbon pollution into the atmosphere - no rules were in effect that limited their emissions of carbon dioxide, the primary driver of global warming. Now, for the first time, the EPA has finalized new rules, or standards, that will reduce carbon emissions from power plants. Known as the Clean Power Plan, these historic standards represent the most significant opportunity in years to help curb the growing consequences of climate change.

The data center will have a total carbon profile, that includes the many different aspects of a data center contributing to carbon emissions. This includes manufacturing, packaging, transportation, storage, operation of the data center, and decommissioning. Thus it is important to notice that we not only need to consider the operation but also the construction and decommission phases.

6.5.2 Data Center Operational Impact

One of the main operational impacts is the cost and emissions of a data center cause by running, and cooling the servers in the data center. Naturally this is dependent on the type of fuel that is used to produce the energy. The actual carbon impact using electricity certainly depends on the type of powerplant that is used to provide it. These energy costs and distribution of where the energy comes from can often be looked up by geographical regions on the internet or form the local energy provider. Municipal government organizations may also have such information. Tools such as the [Indiana State Profile and Energy Use \[7\]](#).

may provide valuable information to derive such estimates.

Correlating a data center with cheap energy is a key factor. To estimate both costs in terms of price and carbon emission Schneider provides a convenient Carbon estimate calculator based on energy consumption.

- <https://www.schneider-electric.com/en/work/solutions/system/s1/data-center-and-network-systems/trade-off-tools/data-center-carbon-footprint-comparison-calculator/tool.html>
- <http://it-resource.schneider-electric.com/digital-tools/calculator-data-center-carbon>

If we calculate the total cost, we need naturally add all costs arising from build and teardown phase as well as operational upgrades.

Exercises

E.Carbon.1: Carbon footprint of a data center

Complete the definitions of the terms used in the relevant section

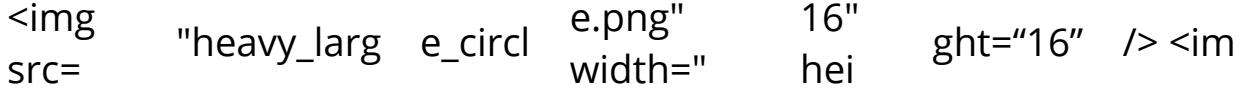
E.Carbon.2: Carbon footprint of data centers

World wide we have many data centers. Your task will be to find the carbon emission of a data center and its cost in \$ based on energy use on a yearly basis. Add your findings to the following table. Make sure you avoid redundant reporting and find a new datacenter. A google doc will be provided to coordinate with the class participants

Table: Cost of the data center

.

Data Center	Location	Year	Electricity Cost*	IT Load	Yearly Cost	Yearly CO2
-------------	----------	------	-------------------	---------	-------------	------------

Center	Cost*	Load Cost	Footprint
			

*as adjusted in calculator

If you find other estimates for a data center or an entire data center fleet such as AWS world wide, please provide citations.

E.Carbon.3: Your own Carbon footprint

It is interesting to compare and measure your own carbon footprint. We will ask you anonymously to report your carbon footprint via a form we will prepare in future. As the time to do this is less than 2 minutes, We ask all students to report their footprint.

Please use the calculator at:

- <http://carbonfootprint.c2es.org/>

6.5.3 Power Usage Effectiveness

One of the frequent measurements in data centers that is used is the Power usage effectiveness or PUE in short. It is a measurement to identify how much energy is used for the computing equipment versus other energy costs such as air conditioning.

Formally we define it as

PUE is the ratio of total amount of energy used by a computer data center facility to the energy delivered to computing equipment.

PUE was published in 2016 as a global standard under [ISO/IEC 30134-2:2016](#).

The inverse of PUE is the data center infrastructure efficiency (DCIE).

The best value of PUE is 1.0. Any data center must be higher than this value as offices and other cost surely will arise when we look at the formula

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}}$$

$$\text{PUE} = 1 + \frac{\text{Non IT Facility Energy}}{\text{IT Equipment Energy}}$$

According to the PUE calculator at

- <https://www.42u.com/measurement/pue-dcie.htm>

The following ratings are given

PUE	DCIS	Level of Efficiency
3.0	33%	Very Inefficient
2.5	40%	Inefficient
2.0	50%	Average
1.5	67%	Efficient
1.2	83%	Very Efficient

PUE is a very popular metric as it is relatively easy to calculate and provides a metric that can easily compare data centers between each other.

This metric comes also with some drawbacks:

- It does not integrate for example climate based differences, such as that the energy use to cool a data center in colder

climates is less than in warmer climates. However, this may actually be a good side-effect as this will likely result in less cooling needs and therefore energy costs.

- It also forces large data centers with many shared servers in contrast to small data centers where operational cost may become relevant.
- It does not take into consideration recycled energy to for example heat other buildings outside of the data center.

Hence it is prudent not to just look at the PUE but also at other metrics that lead to the overall cost and energy usage of the total ecosystem the data center is located in.

Already in 2006, Google reported its six data centers efficiency as 1.21 and Microsoft as 1.22 which at that time were considered very efficient. However over time these targets have shifted and today's data centers achieve much lower values. The Green IT Cube in Darmstadt, Germany even reported 1.082. According to Wikipedia an unnamed Fortune 500 company achieved with 30000 SuperMicro blades a PUE of 1.06 in 2017.

Exercises

E.PUE.1: Lowest PUE you can find

What is the lowest PUE you can find. Provide details about the system as well as the date when the PUE was reported.

6.5.4 Hot-Cold Aisle

To understand hot-cold aisles, one must take a brief foray into the realm of physics and energy. Specifically, understanding how a temperature gradient tries to equalize. The most important formula to know is the heat transfer equation ([Equation 2](#)).

$$q = h_c A(t_a - t_s) \quad (2)$$

Here, q is the amount of heat transferred for a given amount of time. For this example, we'll calculate it as W/hour as that is, conveniently, how energy is billed. Air moving at a moderate speed will transfer approximately 8.47 Watts per Square Foot per Hour. A 1U server is 19 inches wide and about 34 inches deep. Multiplying the two values gives us a cross section of 646 square inches, or 4.48 square feet. Plugging these values into our equation above gives us:

$$q = 8.47 * 4.48 * (t_a - t_s) \quad (3)$$

This begins to point us towards why hot-cold aisles are important. If we introduce cold air from the AC system into the same aisle that the servers are exhausting into, the air will mix and begin to average out. For example, if our servers are producing exhaust at 100F and our AC unit provides 65F at the same rate, then the average air temperature will become 82.5F (assuming balanced air pressure). This has a deleterious effect on our server cooling - warmer air takes heat away from warmer surfaces slower than cooler air:

$$1,328.2 = 8.47 * 4.48 * (100 - 65)$$

$$664.0 = 8.47 * 4.48 * (100 - 82.5)$$

From the above, we can see that a 35 degree delta allows the center to dissipate 1,300 Watts of waste heat from a 1U server while a 17.5 degree delta allows us to only dissipate 664 Watts of energy. If a server is consuming more than 664 Watts, it'll continue to get warmer and warmer until it eventually reaches a temperature differential high enough to create an equilibrium (or reaches a thermal throttle and begins to reduce performance).

To combat this, engineers developed the idea of designating alternating aisles as either hot or cold. All servers in a given aisle are then oriented such that the AC system provides cool air into the cold aisle where it is drawn in by the server which then exhausts it into the hot aisle where the ventilation system removes it from the room. This has the benefit of maximizing the temperature delta between the provided air and the server's processor(s), reducing the amount of

quantity of air that must be provided in order to cool the server and improving overall system efficiency.

See +[Figure 10](#) to understand how the hot-cold isle configuration is setup in a data center.

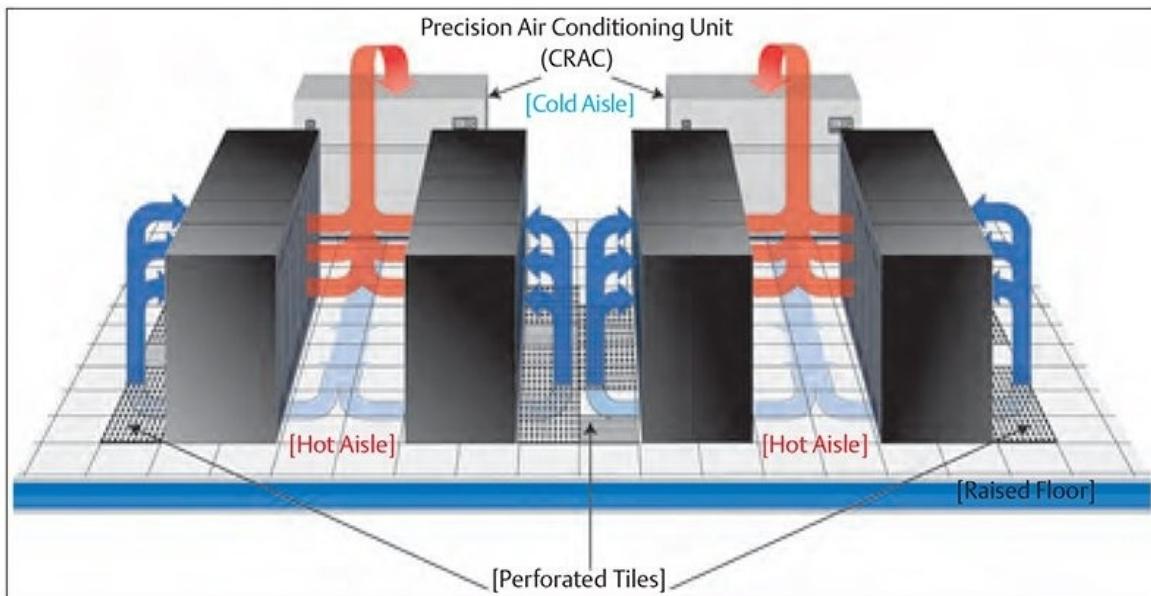


Figure 10: Hot Cold Isle

Source: <https://www.joepowell.com/4-steps-to-better-data-center-cooling/>

6.5.4.1 Containment

While modern data centers employ highly sophisticated mechanisms to be as energy efficient as possible. One such mechanism which can be seen as a improvement on top of the Hot-Cold isle arrange is to use either hot isle containment or cold isle containment. Using a containment system can remove the issue with free flowing air.

As the name somewhat implies in cold air containment, the data centers is designed so that only cold air goes into the cold isle, this makes sure that the system only draws in cold air for cooling purposes. Conversely in hot isle containment design, the hot isle is contained so that the hot air collected in the hot isle is drawn out by

the cooling system and so that the cold air does not flow into the hot isles.

Source:

<https://www.datacenterknowledge.com/archives/2012/11/08/approach-to-data-center-containment>

6.5.4.1.1 Water Cooled Doors

Alternatively, or in addition 

6.5.5 Workload Monitoring

6.5.5.1 Workload of HPC in the Cloud

Clouds and especially university data centers do not just provide virtual machines but provide traditional super computer services. This includes the NSF sponsored XSEDE project. As part of this project the "XDMoD auditing tool provides, for the first time, a comprehensive tool to measure both utilization and performance of high-end cyberinfrastructure (CI), with initial focus on XSEDE. Several case studies have shown its utility for providing important metrics regarding resource utilization and performance of TeraGrid/XSEDE that can be used for detailed analysis and planning as well as improving operational efficiency and performance. Measuring the utilization of high-end cyberinfrastructure such as XSEDE helps provide a detailed understanding of how a given CI resource is being utilized and can lead to improved performance of the resource in terms of job throughput or any number of desired job characteristics.

Detailed historical analysis of XSEDE usage data using XDMoD clearly demonstrates the tremendous growth in the number of users, overall usage, and scale" [8].

Having access to a detailed metrics analysis allows users and center administrators, as well as project managers to better evaluate the use and utilization of such large facilities justifying their existence (see

Figure 11)

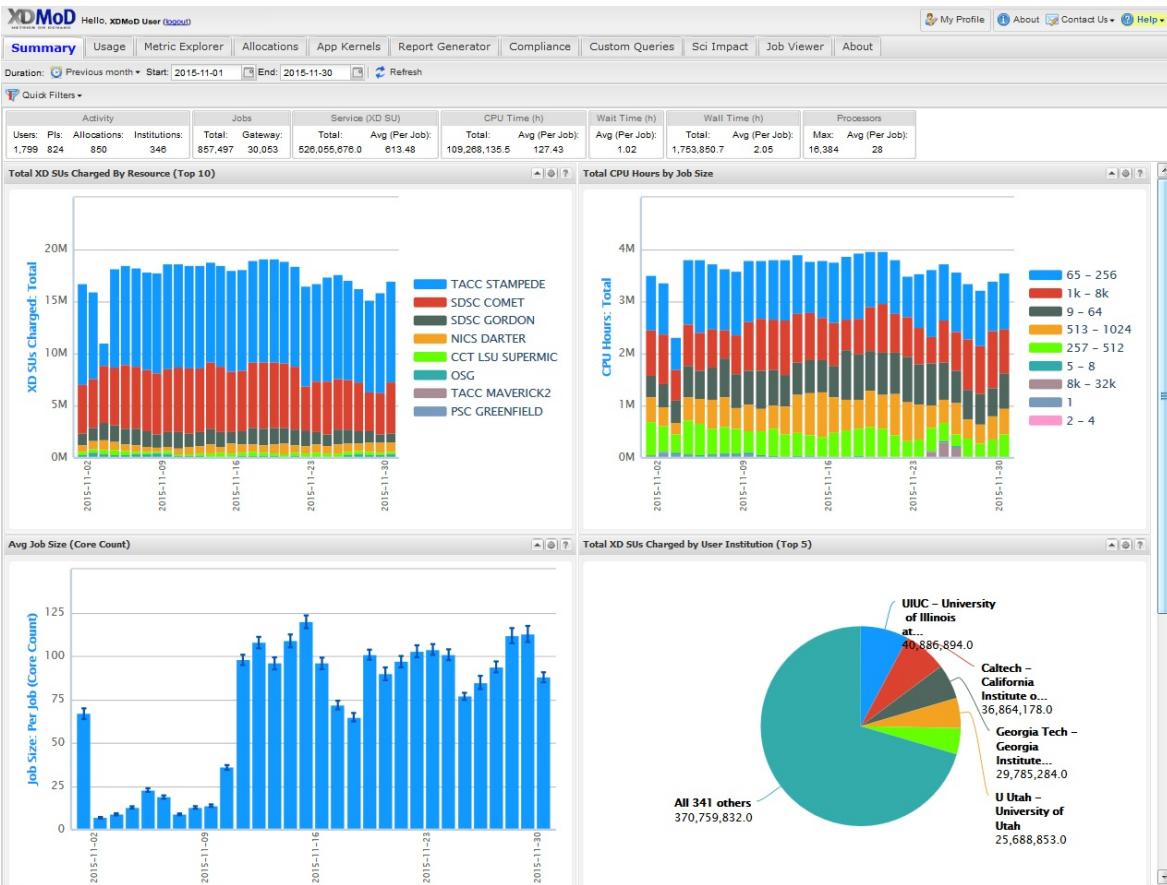


Figure 11: XDMoD: XSEDE Metrics on Demand

Additional information is available at

- <https://open.xdmod.org/7.5/index.html>

6.5.5.2 Scientific Impact Metric

Gregor von Laszewski and Fugang Wang are providing a scientific impact metric to XDMoD and XSEDE. It is a framework that (a) integrates publication and citation data retrieval, (b) allows scientific impact metrics generation at different aggregation levels, and (c) provides correlation analysis of impact metrics based on publication and citation data with resource allocation for a computing facility. This framework is used to conduct a scientific impact metrics evaluation of XSEDE, and to carry out extensive statistical analysis correlating XSEDE allocation size to the impact metrics aggregated by project and Field of Science. This analysis not only helps to provide an

indication of XSEDE'S scientific impact, but also provides insight regarding maximizing the return on investment in terms of allocation by taking into account Field of Science or project based impact metrics. The findings from this analysis can be utilized by the XSEDE resource allocation committee to help assess and identify projects with higher scientific impact. Through the general applicability of the novel metrics we invented, it can also help provide metrics regarding the return on investment for XSEDE resources, or campus based HPC centers [9].

6.5.5.3 Clouds and Virtual Machine Monitoring

Although no longer in operation in its original form [FutureGrid](#) [10] has pioneered the extensive monitoring and publication of its virtual machine and project usage. We are not aware of a current system that provides this level of detail as so far yet. However, efforts as part of XSEDE within the XDMoD project are under way at this time but are not integrated.

Futuregrid provided access to all virtual machine information, as well as usage across projects. An archived portal view is available at [FutureGrid Cloud Metrics](#) [10].

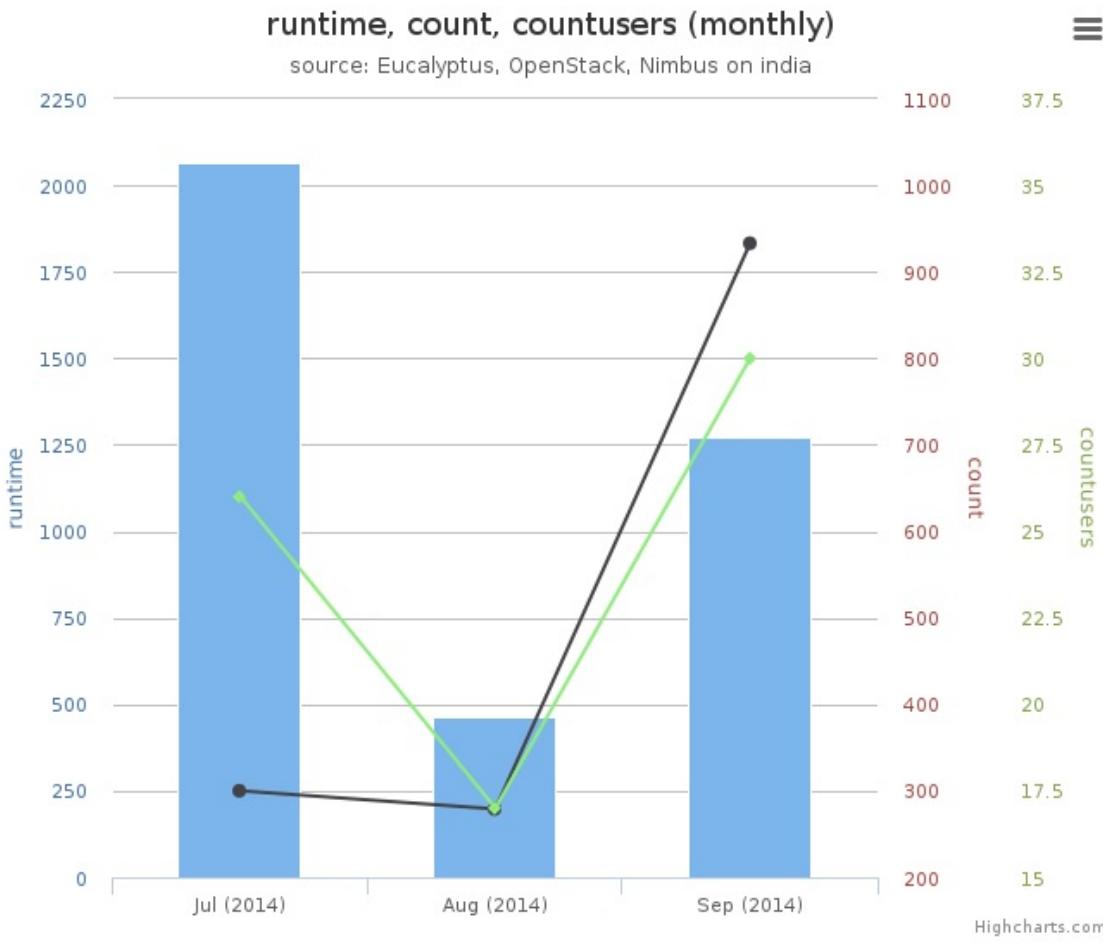


Figure 12: FutureGrid Cloud Metric

Futuregrid offered multiple clouds including clouds based on OpenStack, Eucalyptus, and Nimbus. Nimbus and Eucalyptus are systems that are no longer used in the community. Only OpenStack is the only viable solution in addition to the cloud offerings by Comet that do not uses OpenStack (see [Figure 12](#)).

Futuregrid, could monitor all of them and published its result in its Metrics portal. Monitoring the VMs is an important activity as they can identify VMs that may no longer be used (the user has forgotten to terminate them) or too much usage of a user or project can be detected in early stages.

We like to emphasize several examples where such monitoring is helpful:

- Assume a student participates in a class, metrics and logs allow to identify students that do not use the system as asked for by the instructors. For example it is easy to identify if they logged on and used VMs. Furthermore the length of running a VM ba
- Let us assume a user with willful ignorance does not shut down VMs although they are not used because research clouds are offered to us for free. In fact, this situation happened to us recently while using another cloud and such monitoring capacities were not available to us (on jetstream). The user used up simple handedly the entire allocation that was supposed to be shared with 30 other users in the same project. All accounts of all users were quasi deactivated as the entire project they belonged to were deactivated. Due to allocation review processes it took about 3 weeks to reactivate full access. sed on the tasks to be completed can be compared against other student members.
- In commercial clouds you will be charged money. Therefore, it is less likely that you forget to shutdown your machine
- In case you use github carelessly and post your cloud passwords or any other passwords in it, you will find that within five minutes your cloud account will be compromised. There are individuals on the network that cleverly mine github for such security lapses and will use your password if you indeed have stored them in it. In fact github's deletion of a file does not delete the history, so as a non expert deleting the password form github is not sufficient. You will have to either delete and rewrite the history, but definitely in this case you will need to reset the password. Monitoring the public cloud usage in the data center is important not only in your region but other regions as the password is valid also there and intruders could hijack and start services in regions that you have never used.

In addition to FutureGrid, we like to point out Comet (see other sections). It contains an exception for VM monitoring as it uses a regular batch queuing system to manage the jobs. Monitoring of the

jobs is conducted through existing HPC tools

6.5.5.4 Workload of Containers

Monitoring tools for containers such as for kubernetes are listed at:

- <https://kubernetes.io/docs/tasks/debug-application-cluster/resource-usage-monitoring/>

Such tools can be deployed alongside kubernetes in the data center, but will likely have restrictions to its access. They are for those who operate such services for example in kubernetes. We will discuss this in future sections in more detail.

6.6 EXAMPLE DATA CENTERS

In this section we will be giving some data center examples while looking at some of the major cloud providers.

6.6.1 AWS

AWS focuses on security aspects of their data centers that include four aspects [11]:

- [Perimeter Layer](#)
- [Infrastructure Layer](#)
- [Data Layer](#)
- [Environmental Layer](#)

The [global infrastructure](#) [12] as of January 2019 includes 60 Availability Zones within 20 geographic Regions. Plans exist to add 12 Availability Zones and four additional Regions in Bahrain, Hong Kong SAR, Sweden, and a second AWS GovCloud Region in the US (see [Figure 13](#)).

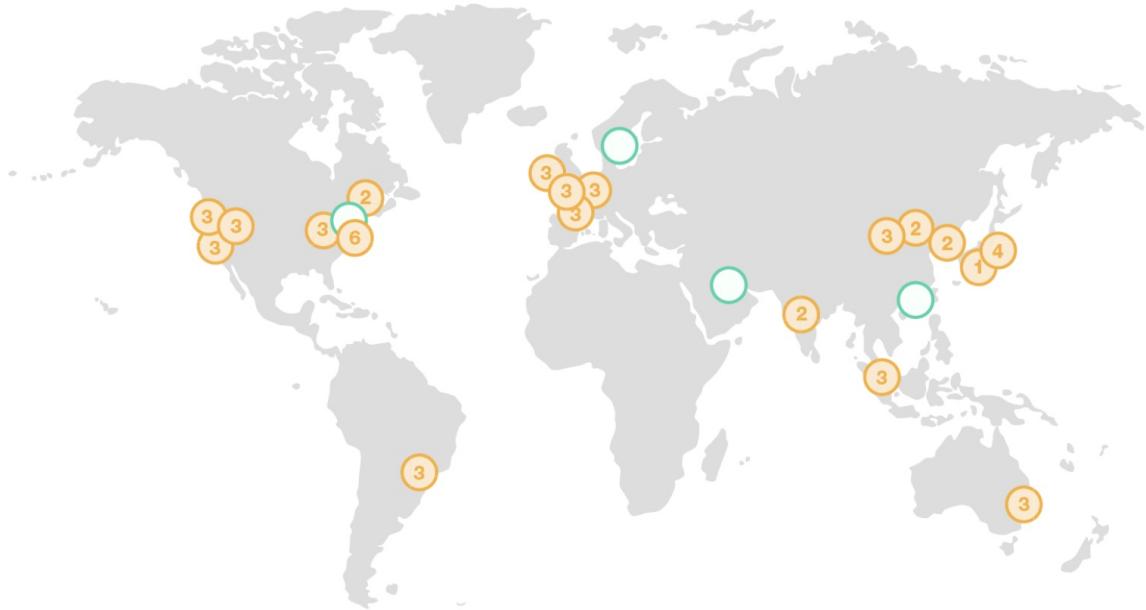


Figure 13: AWS regions [12]

Amazon strives to achieve high availability through multiple availability zones, improved continuity with replication between regions, meeting compliance and data residency requirements as well as providing geographic expansion. See [Figure 14](#)

The regions and number of availability zones are as follows:

- Region US East: N. Virginia (6), Ohio (3) US West N. California (3), Oregon (3)
- Region: Asia Pacific Mumbai (2), Seoul (2), Singapore (3), Sydney (3), Tokyo (4), Osaka-Local (1)1 Canada Central (2) China Beijing (2), Ningxia (3)
- Region: Europe Frankfurt (3), Ireland (3), London (3), Paris (3) South America São Paulo (3)
- Region Gov Cloud: AWS GovCloud (US-West) (3)
- New Region (coming soon): Bahrain, Hong Kong SAR, China, Sweden, AWS GovCloud (US-East)

6.6.2 Azure

Azure claims to have more global [regions](#) [13] than any other cloud provider. They motivate this by their advertisement to bring and

applications to the users around the world. The goal is similar as other commercial hyperscale providers by introducing preserving data residency, and offering comprehensive compliance and resilience. As of Aug 29, 2018 Azure supports 54 regions worldwide. These regions can currently be accessed by users in 140 countries (see [Figure 14](#)). Not every service is offered in every region as the service to region matrix shows:

- <https://azure.microsoft.com/en-us/global-infrastructure/services/>



Figure 14: Azure regions [[13](#)]

6.6.3 Google

From [Google](#) [[14](#)] we find that on Aug. 29th Google has the following data center locations (see [Figure 15](#)):

- **North America:** Berkeley County, South Carolina; Council Bluffs, Iowa; Douglas County, Georgia; Jackson County, Alabama; Lenoir, North Carolina; Mayes County, Oklahoma; Montgomery County, Tennessee; The Dalles, Oregon
- **South America:** Quilicura, Chile
- **Asia:** Changhua County, Taiwan; Singapore
- **Europe:** Dublin, Ireland; Eemshaven, Netherlands; Hamina, Finland; St Ghislain, Belgium



Figure 15: Google data centers [14]

Each data center is advertised with a special environmental impact such as a unique cooling system, or wildlife on premise. Google's data centers support its service infrastructure and allow hosting as well as other cloud services to be offered to its customers.

Google highlights its efficiency strategy and methods here:

- <https://www.google.com/about/datacenters/efficiency/>

They summarize their offers are based on

- Measuring the PUE
- Managing airflow
- Adjusting the temperature
- Use free Cooling
- Optimizing the power distribution

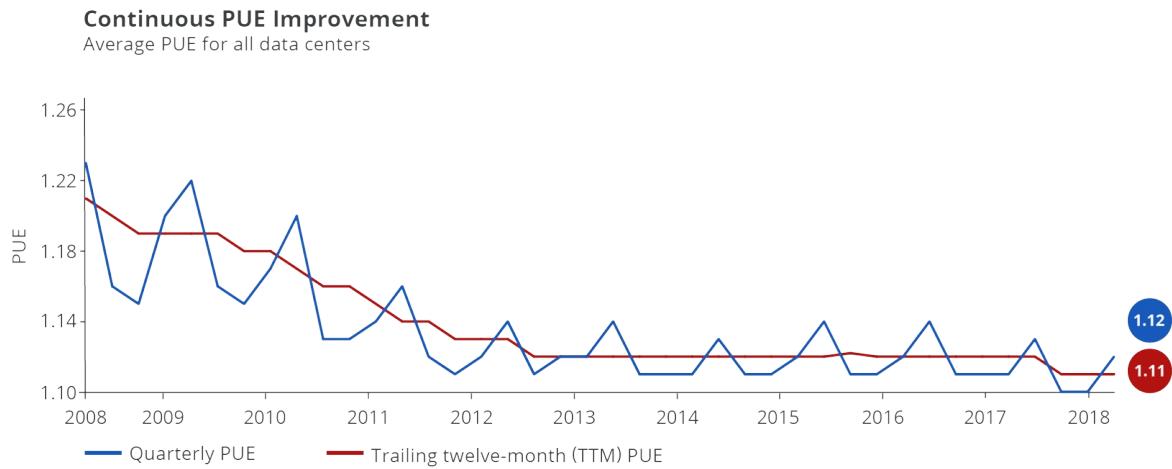


Figure 16: PUE data for all large-scale Google data centers

The [PUE \[15\]](#) data for all large-scale Google data centers is shown in [Figure 16](#)

An important lesson from Google is the PUE boundary. That is the different efficiency based on the closeness of the IT infrastructure to the actual data center building. This indicates that it is important to take at any providers definition of PUE in order not to report numbers that are not comparable between other vendors and are all encompassing.

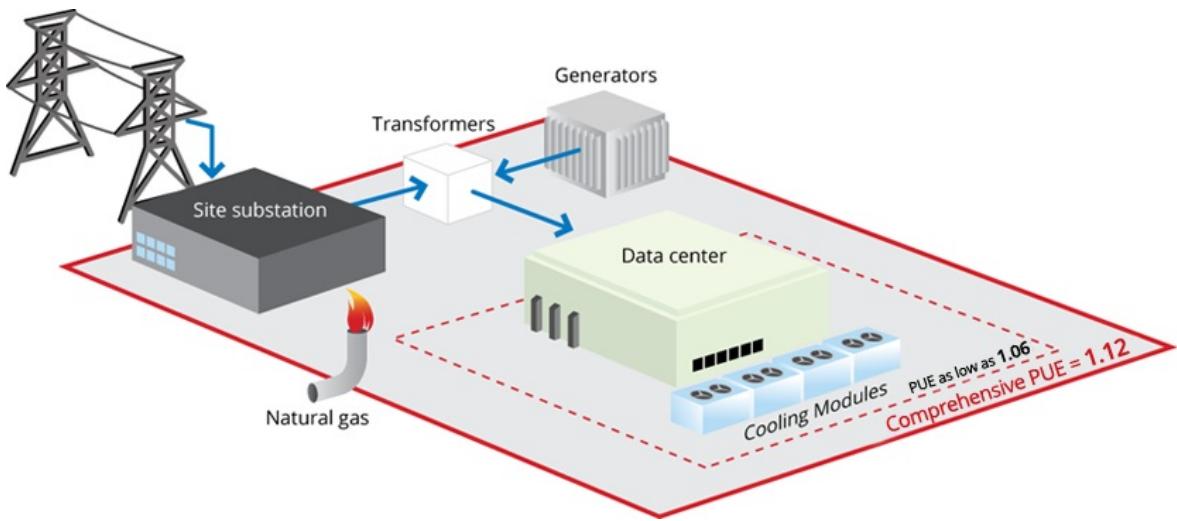


Figure 17: Google data center PUE measurement boundaries [\[15\]](#)

[Figure 17](#) shows the Google data center PUE measurement boundaries. The [average PUE \[15\]](#) for all Google data centers is 1.12, although we could boast a PUE as low as 1.06 when using narrower boundaries.

As a consequence, Google is defining its PUE in detail in [Equation 4](#).

$$PUE = \frac{ESIS + EITS + ETX + EHV + ELV + EF}{EITS - ECRAC - EUPS - ELV + ENet1} \quad (4)$$

where the abbreviations stand for

- ESIS = Energy consumption for supporting infrastructure power substations feeding the cooling plant, lighting, office space, and some network equipment
- EITS = Energy consumption for IT power substations feeding servers, network, storage, and computer room air conditioners (CRACs)
- ETX = Medium and high voltage transformer losses
- EHV = High voltage cable losses
- ELV = Low voltage cable losses
- EF = Energy consumption from on-site fuels including natural gas & fuel oils
- ECRAC = CRAC energy consumption
- EUPS = Energy loss at uninterruptible power supplies (UPSes) which feed servers, network, and storage equipment
- ENet1 = Network room energy fed from type 1 unit substitution

For more [details](#) see [15].

6.6.4 IBM

IBM maintains almost 60 data centers, which are placed globally in 6 regions and 18 availability zones. IBM targets businesses while offering local access to its centers to allow for low latency. IBM states

that through this localization users can decide where and how data and workloads and address availability, fault tolerance and scalability. As IBM is business oriented it also stresses its certified security.

More information can be obtained from:

- <https://www.ibm.com/cloud/data-centers/>

A special service offering is provided by Watson.

- <https://www.ibm.com/watson/>

which is focusing on AI based services. It includes PaaS services for deep learning, but also services that are offered to the healthcare and other communities as SaaS

6.6.5 XSEDE

XSEDE is an NSF sponsored large distributed set of clusters, supercomputers, data services, and clouds, building a “single virtual system that scientists can use to interactively share computing resources, data and expertise”. The Web page of XSEDE is located at

- <https://www.xsede.org/>

Primary compute resources are listed in the resource monitor at

- <https://portal.xsede.org/resource-monitor>

For cloud Computing the following systems are of especial importance although selected others may also host container based systems while using singularity (see [Figure 18](#)):

- Comet virtual clusters
- Jetstream OpenStack

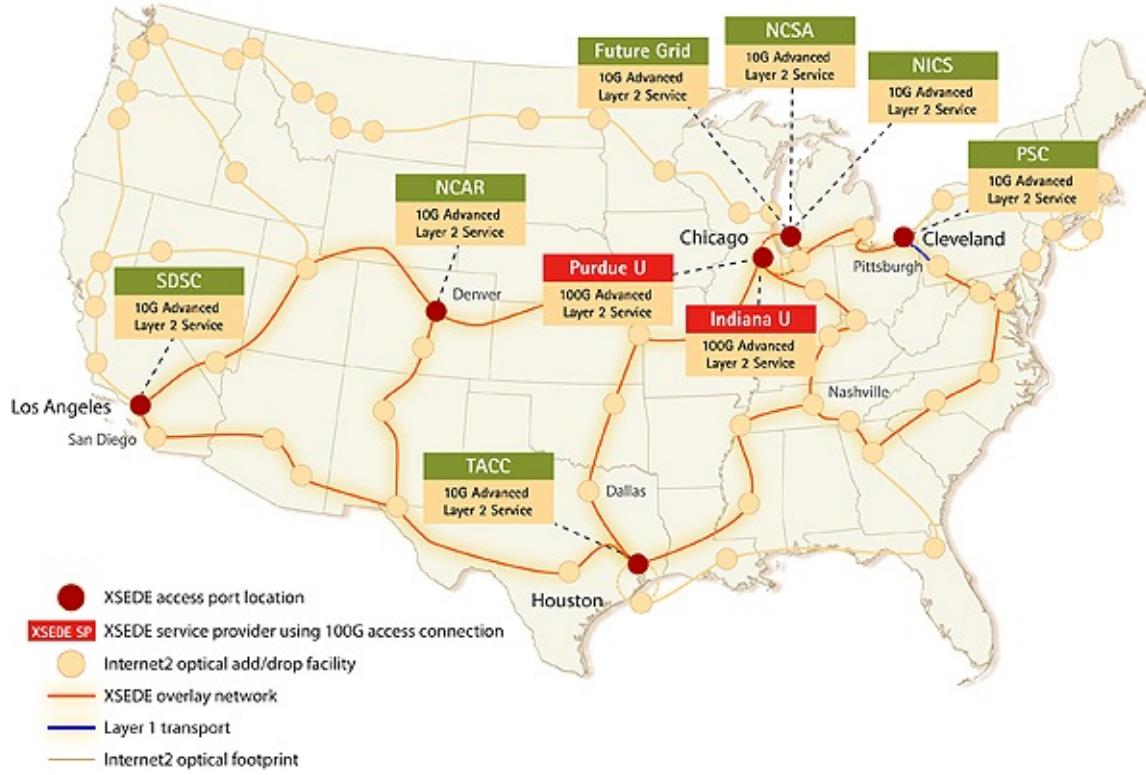


Figure 18: XSEDE distributed resource infrastructure

6.6.5.1 Comet

The comet machine is a larger cluster and offers bare metal provisioning based on KVM and SLURM. Thus it is a unique system that can run at the same time traditional super computing jobs such as MPI based programs, as well as jobs that utilize virtual machines. With its availability of >46000 cores it provides one of the largest NSF sponsored cloud environment. Through its ability to provide bare metal provisioning and the access to infiniband between all virtual machines it is an ideal machine for exploring performance oriented virtualization techniques.

Comet has about 3 times more cores than Jetstream.

6.6.5.2 Jetstream

Jetstream is a machine that specializes in offering a user friendly cloud environment. It utilizes an environment called atmosphere that

is targeting inexperienced scientific cloud users. It also offers an OpenStack environment that is used by atmosphere and is for classes such as ours the preferred access method. More information about the system can be found at

- <https://dcops.iu.edu/>

6.6.6 Chameleon Cloud

Chameleon cloud is a configurable experimental environment for large-scale cloud research. It is offering OpenStack as a service including some more advanced services that allow experimentation with the infrastructure.

- <https://www.chameleoncloud.org/>

An overview of the hardware can be obtained from

- <https://www.chameleoncloud.org/hardware/>

6.6.7 Indiana University

Indiana University has a data center in which many different systems are housed. This includes not only jetstream, but also many other systems. The systems include production, business, and research clusters and servers. See [Figure 19](#)



Figure 19: IU Data Center

On the research cluster side it offers Karst:

- <https://kb.iu.edu/d/bezu>

One of the special systems located in the data center and managed by the Digital Science Center is called Futuresystems, which provides a great resource for the advanced students of Indiana University focusing on data engineering. While systems such as Jetstream and Chameleon cloud specialize in production ready cloud environments, Futuresystems, allows the researchers to experiment with state-of-the-art distributed systems environments supporting research. It is available with Comet and thus could also serve as an on-ramp to using larger scale resources on comet while experimenting with the setup on Futuresystems.

Such an offering is logical as researchers in the data engineering track want to further develop systems such as Hadoop, SSpark, or container based distributed environments and not use the tools that are released for production as they do not allow improvements to the infrastructure. Futuresystems is managed and offered by the Digital Science Center.

Hence IU offers very important but needed services

- Karst for traditional supercomputing
- Jetstream for production use with focus on virtual machines
- Futuresystems for state-of-the-art research experiment environments with access to bare metal.

6.6.8 Shipping Containers

A few years ago data centers build from shipping containers were very popular. This includes several main Cloud providers. Such providers have found that they are not the best way to develop centers at scale. This includes [Microsoft](#) and [Google](#). The current trend however is to build mega or hyperscale data centers.

6.7 SERVER CONSOLIDATION

One of the driving factors in cloud computing and the rise of large scale data centers is the ability to use server virtualization to place more than one server on the same hardware. Formerly the services were hosted on their own servers. Today they are managed on the same hardware although they look to the customer like separate servers.

As a result we find the following advantages:

- **reduction of administrative and operations cost:** While we reduce the number of servers and utilize hardware to host multiple on them management cost, space, power, and maintenance cost are reduced.
- **better resource utilization:** Through load balancing strategies servers can be better utilized while for example increase load so resource idling is avoided.
- **increased reliability:** As virtualized servers can be snapshotted, and mirrored, these features can be utilized in strategies to increase reliability in case of failure.
- **standardization:** As the servers are deployed in large scale, the infrastructure is implicitly standardized based on server, network, and disk, making maintenance and replacements easier. This also includes the software that is running on such servers (OS, platform and may even include applications).

6.8 DATA CENTER IMPROVEMENTS AND CONSOLIDATION

Due to the immense number of servers in data centers, as well as the increased workload on its servers, the energy consumption of data centers is large not only to run the servers, but to provide the necessary cooling. Thus it is important to revisit the impact such data centers have on the energy consumption. One of the studies that looked into this is from 2016 and is published by [LBNL \[16\]](#) In this study the data center electricity consumption back to 2000 is

analyzed while using previous studies and historical shipment data. A forecast is with different assumption is contrasts till 2020

Figure Energy Forecast depicts “an estimate of total U.S. data center electricity use (servers, storage, network equipment, and infrastructure) from 2000-2020” (see [Figure 20](#)).

While in “2014 the data centers in the U.S. consumed an estimated 70 billion kWh” or “about 1.8% of total U.S. electricity consumption”. However, more recent studies find an increase by about 4% from 2010-2014. This contrasts a large derivation from the 24% that were originally predicted several years ago. The study finds that the predicted energy use would be approximately 73 billion kWh in 2020.

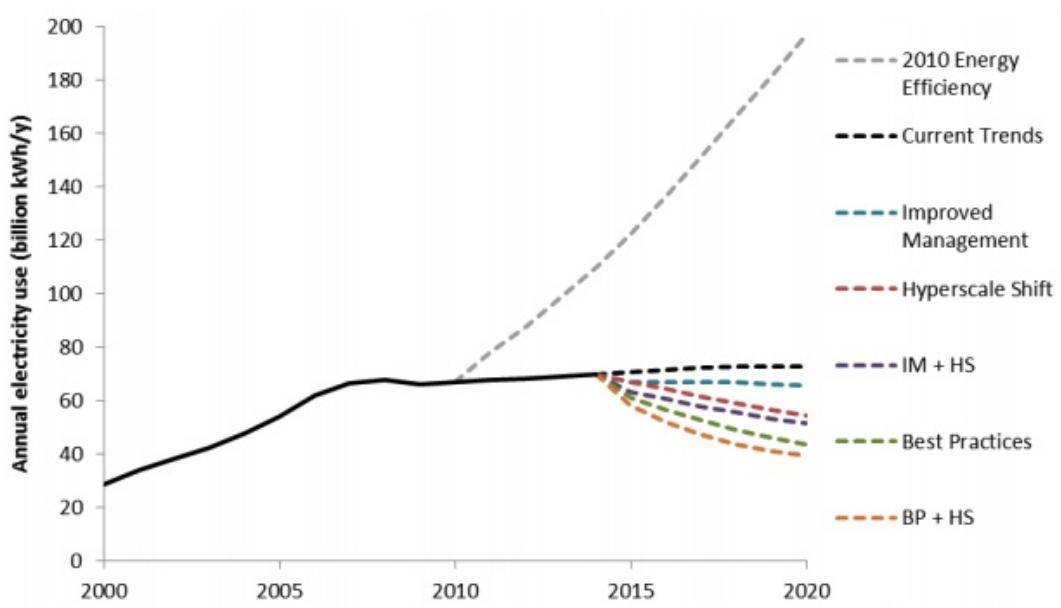


Figure 20: Energy Forecast [\[16\]](#)

It is clear that the original prediction of large energy consumption motivated a trend in industry to provide more energy efficient data centers. However if such energy efficiency efforts would not be conducted or encouraged we would see a completely different scenario.

The scenarios are identified that will significantly impact the

prediction:

- **improved management** increases energy-efficiency through operational or technological changes with minimal investment. Strategies include improving the least efficient components.
- **best practices** increases the energy-efficiency gains that can be obtained through the widespread adoption the most efficient technologies and best management practices applicable to each data center type. This scenario focuses on maximizing the efficiency of each type of data center facility.
- **hyperscale data centers** where the infrastructure will be moved from smaller data centers to larger hyperscale data centers.

6.9 PROJECT NATICK

To reduce energy consumption in data centers and reduce cost of cooling Microsoft has developed Project Natick. To tackle this problem Microsoft has built underwater datacenter. Another benefit of this project is that data center can be deployed in large bodies of water to serve customers residing in that area so it helps to reduce latency by reducing distance to users and therefore increasing data transfer speed. There are two phases of this project.

The project was executed in two phases.

Phase 1 was executed between August to November 2015. In this phase Microsoft was successfully able to deploy and operate vessel underwater. The vessel was able to tackle cooling issues and effect of biofouling as well. Biofouling is referred to as the fouling of pipes and underwater surfaces by organisms such as barnacles and algae.

The PUE (Power Usage Effectiveness) of Phase1 vessel was 1.07 which is very efficient and a perfect WUE (Water Usage Effectiveness) of

exactly 0, while land data centers consume ~ 4.8 liters of water per KWH. This vessel consumed computer power equivalent to 300 Desktop PCs and was of 38000 lbs and it operated for 105 days.



Figure 21: The Leona Philpot prototype

See [Figure 21](#), The Leona Philpot prototype was deployed off the central coast of California on Aug. 10, 2015. Source: [Microsoft \[17\]](#)

The phase 2 started in June 2018 and lasted for 90 days. Microsoft deployed a vessel at the European Marine Energy Center in UK.

The phase 2 vessel was 40ft long and had 12 racks containing 864 servers. Microsoft powered this data center using 100% renewable energy. This data center is claimed to be able to operate without maintenance for 5 years. For cooling Microsoft used infrastructure which pipes sea water through radiators in back on server racks and then move water back in to ocean.

The total estimated lifespan of a Natick datacenter is around 20 years, after which it will be retrieved and recycled.



Figure 22: The Northern Isles prototype

Source: [Microsoft \[18\]](#)

[Figure 22](#) shows the Northern Isles prototype being deployed near Scotland.

Although the cooling provides a significant benefit while using seawater, it is clear that long time studies need to be conducted with this approach and not just studies over a very short period of time. The reason for this is not just the measurement of a PUE factor or the impact of algae on the infrastructure, but also how such a vessel impacts the actual ecosystem itself.

Some thought on this include:

1. How does the vessel increase the surrounding water temperature and effects the ecosystem
2. If placed in saltwater, corrosion can occur that may not only effect the vessel, but also the ecosystem
3. Positive effects could also be created on ecosystems, which is for example demonstrated by creation of artificial reefs. However if the structure has to be removed after 20 years, what impact has it on the ecosystem.

Find more about this at [19]

6.10 RENEWABLE ENERGY FOR DATA CENTERS

Explain the principal and showcase some examples:

- Solar: <https://9to5google.com/2019/01/17/largest-ever-solar-farms-google/>
- Wind: <https://www.datacenterknowledge.com/wind-powered-data-centers>
- Hydro: <http://www.hydroquebec.com/data-center/advantages/clean-energy.html> there will be others
- Thermal: find better resource <https://spectrum.ieee.org/energywise/telecom/internet/iceland-data-center-paradise>
- Recyclers: <https://www.datacenterknowledge.com/data-centers-that-recycle-waste-heat>

What other aspects exist:

Energy Storage:

- Batteries
- Store energy in other forms

6.11 SOCIETAL SHIFT TOWARDS RENEWABLES

The data center as example.

Government efforts to support renewable in benefit of the society:

- Germany
- China
- Island
- Corporations: Google, AWS, IBM, ...

Also look at the US state of California

Everyone is doing it.

See also

- https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2018/Jan/IRENA_2017

6.12 EXERCISES

E.Energy.1:

Pick a renewal energy from [Section 6.10](#) and describe what it is. Find data centers that use this energy form. Create a section and contribute it to the datacenter.md file.

E.Energy.2:

Pick a country, state, or company from [Section 6.11](#) and summarize their efforts towards renewable energy and impacts within the society. Create a section and contribute it to the datacenter.md file.



◆ Learning Objectives

- Review classical architectural models leading up to cloud computing.
- Review some major cloud architecture views.
- Visualize the NIST cloud architecture
- Discuss an architecture for multicloud frameworks.

While we have introduced in our introductory section a number of definitions for cloud computing, as well as an architectural view for clouds based on the as a Service model, we will look a bit closer at other alternative views. These views are in some cases important as they provide appropriate abstractions for more detailed implementations.

7.1 EVOLUTION OF COMPUTE ARCHITECTURES

We start our observation with some depiction of some of the important architectural models motivating the current state of information technology services we provide in [Figure 23](#). The [original figure](#) has been updated by von Laszewski to include the mobile computing and the internet of things phase that is bringing rapid changes to how we perceive and use the cloud in the near future.

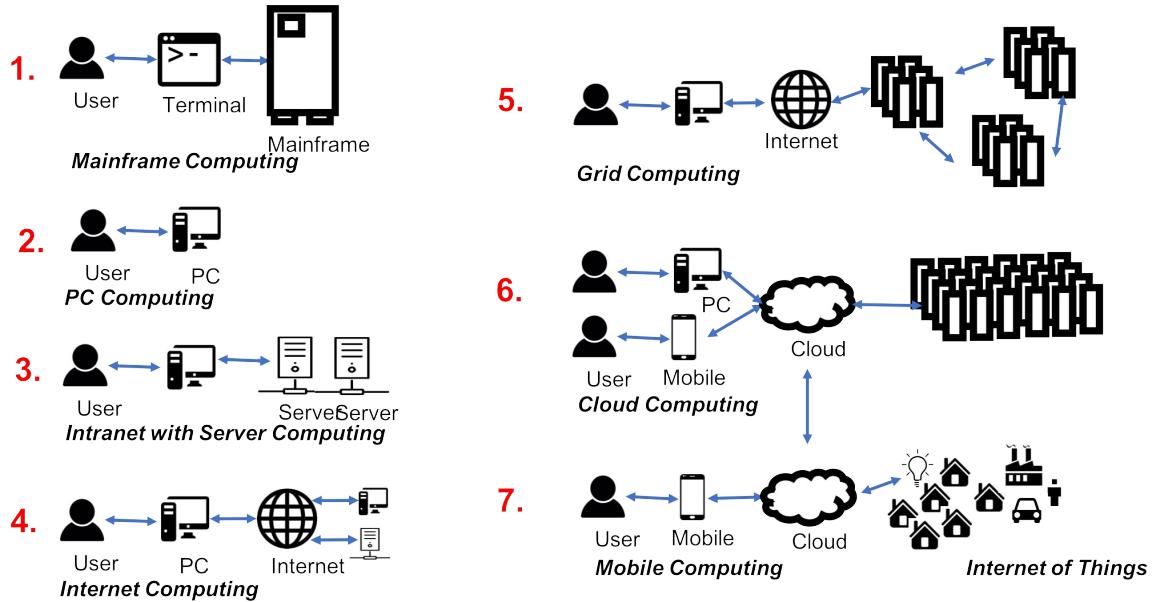


Figure 23: Evolution of Compute Architectures

We define the following terminology based on the evolution of compute architectures:

7.1.1 Mainframe Computing

Mainframe computing is using the larger and more reliable computers, like IBM System z9, to run the critical applications, bulk data processing, enterprise resource planning and business transaction processes.

Mainframe Computing refers to

- This section can be contributed by a student

7.1.2 PC Computing

The term PC is short for personal computer. The first PCs were introduced by IBM to the market. PCs need an operating system such as Windows, macOS, or Linux

PC Computing refers to

an era where consumers predominantly used personal

computers to conduct their work. Such computers were mostly stand alone without network as early networks were not available to consumers.

7.1.3 Intranet and Server Computing

We refer to Intranet and Server Computing as an environment in which

the computers are part of an private network, also called, intranet, that is contained within an enterprise and later on also homes. Intranets are able to connect many local resources within a Local but also a wide area network

7.1.4 Grid Computing Computing

and its evolution is defined in [The Grid-Idea and Its Evolution](#). The original definition of Grid computing has been summarise as follows:

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. [20]

However, in the paper we also define that Grids were not just about computing, but introduced an approach that through the introduction of virtual organizations lead to the following definition

A production Grid is a shared computing infrastructure of hardware, software, and knowledge resources that allows the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations to enable sophisticated international scientific and business-oriented collaborations.

This definition is certainly including services that are today offered by the Cloud. Hence in the early days of cloud computing there was a

large debate occurring if cloud is just another term for Grid. In [Cloud Computing and Grid Computing 360-Degree Compared] (http://datasys.cs.iit.edu/publications/2008_GCE08_Clouds_Grids.pdf) an analysis is conducted between the different architecture models outlining that collective resources and connectivity protocols introduced by the Grid community have been replaced by the cloud with platform and unified resources.

To provide a very simple but possibly incomplete comparison, Cloud computing integrated infrastructure such as supercomputers and other large scale resources through unified protocols. The effort was initially provided by research institutions but have been introduced in business. However, with the growth of the data centers to foster common tasks such as Web hosting, we see a clear difference:

- while the Grid was originally designed to give a few scientist access to the biggest agglomerated research supercomputers,
- business focused on serving originally millions of users with the need to run only a few data or compute services.

This certainly resulted in independent development, while cloud computing has today consumed Grids. Tools such as the GLobus toolkit are no longer widely used, and the development has shifted to the support of data services only.

7.1.5 Internet Computing

With the occurrence of the WWW protocols, internet computing brought to the consumers a global computer network providing a variety of information and communication facilities.

Internet Computing refers to

the infrastructure that enables sharing of data, within the WWW community.

Internet computing also comprises early infrastructures such as AOL,

which popularized the term you got mail

7.1.6 Cloud Computing

Cloud Computing refers to

- A written section can be contributed by student

We have provided a lecture about the definition of cloud computing previously

7.1.7 Mobile Computing

Mobile Computing refers to

a diverse set of devices allowing users to access data and information from wherever they are with mobile devices such as cell phones or tablet computers. mobile computing is dominated by transmission of data, voice, and video over a network via the mobile device

7.1.8 Internet of Things Computing

Internet of Things Computing refers to

devices that are interconnected via the internet while they are embedded in things or common objects. The devices send and receive data to be integrated into a network with sensors and actuators reacting upon sensory and other data.

7.1.9 Edge Computing

In addition, we need to point out two additional terms that we will integrate in this image. Edge Computing and Fog Computing. Currently there is still some debate about what these terms are, but we will follow the following definitions:

Edge Computing refers to

computing conducted on the very edge of infrastructure. This means that data that is not needed in the data center can be calculated and analyzed on the edge devices instead. No interaction between cloud services is needed. Only the absolute required data is sent to the cloud.

7.1.10 Fog Computing

FoG Computing refers to

computing conducted in-between the cloud and the edge devices. This could be for example part of a smart network, that hosts a small set of analytics capabilities, so that the data does not have to travel back to the data center, but the edge device is not powerful enough to do the calculation. Thus a Fog computing infrastructure provides the ability to conduct the analysis closer to the edge saving valuable resources while not needing to transmit all data to the data center although it will be analyzed

7.2 As a Service Architecture Model

The as a Service architecture was one of the earliest definition of cloud architecture while focussing on the service aspect provided by the cloud. The layers such as IaaS, PaaS, and SaaS provide a layered architecture view while separating infrastructure, platform, and services. This allows a separation of concerns typically between infrastructure providers, platform developers, and software architects using platforms and or infrastructure services.

The typical triangular diagram (see [Figure 24](#)) is often used to represent it.

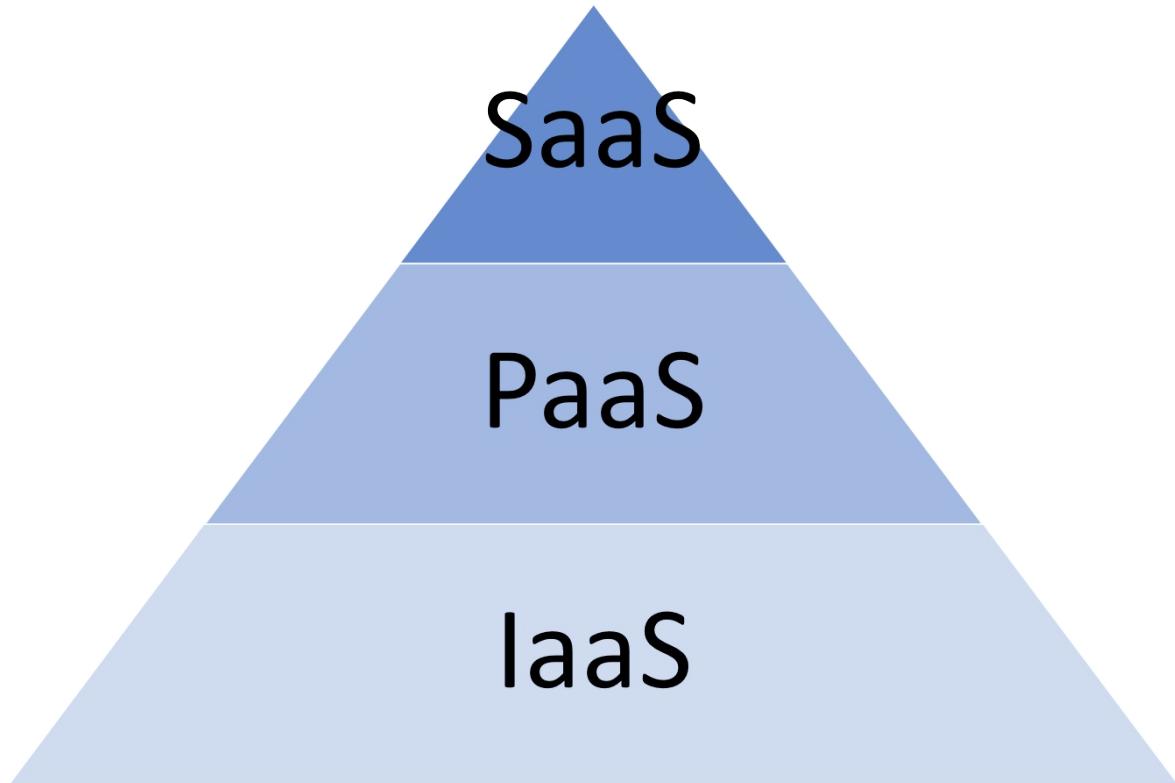


Figure 24: Infrastructure as a Service

7.3 PRODUCT OR FUNCTIONAL BASED MODEL

When we inspect prominent providers such as Amazon, Azure, and Google, we find that on their Web pages they do provide their customers an alternative view that is motivated by exposing numerous products to the customers grouped by functions. These services are often in the hundreds. To achieve the exposure of the products in a meaningful fashion, they introduce a functional view motivation a functional architecture view of the cloud.

When we analyse these functions for example for Amazon Web services we find the following

- Compute
- Storage
- Databases
- Migration
- Networking & Content Delivery

- Developer Tools
- Management Tools
- Media Services
- Security, Identity & Compliance
- Machine Learning
- Analytics
- Mobile
- Augmented reality and Virtual Reality
- Application Integration
- Customer Engagement
- Business Productivity
- Desktop & App Streaming
- Internet of Things
- Game Development
- AWS Marketplace Software
- AWS Cost Management

From this we derive that for the initial contact to the customer the functionality is put in foreground, rather than the distinction between SaaS, PaaS, and IaaS. If we sort these services into the as a Service mode we find:

- IaaS
 - Compute
 - Storage
 - Databases
 - Migration
 - Networking & Content Delivery
- PaaS
 - Developer Tools
 - Management Tools
 - Media Services
 - Security, Identity & Compliance
 - Machine Learning
 - Analytics
 - Mobile

- Augmented reality and Virtual Reality
- Application Integration
- Customer Engagement
- Business Productivity
- Desktop & App Streaming
- Game Development
- AWS Marketplace Software
- AWS Cost Management
- Internet of Things

We observe that AWS focusses on providing infrastructure and platforms so others can provide integrated service to its customers.

Other examples for product lists such as the one from Azure are provided in the Appendix.

7.4 NIST CLOUD ARCHITECTURE

In the introduction we have extensively discussed the NIST cloud architecture. A Nice visual representation is provided in [Figure 25](#).

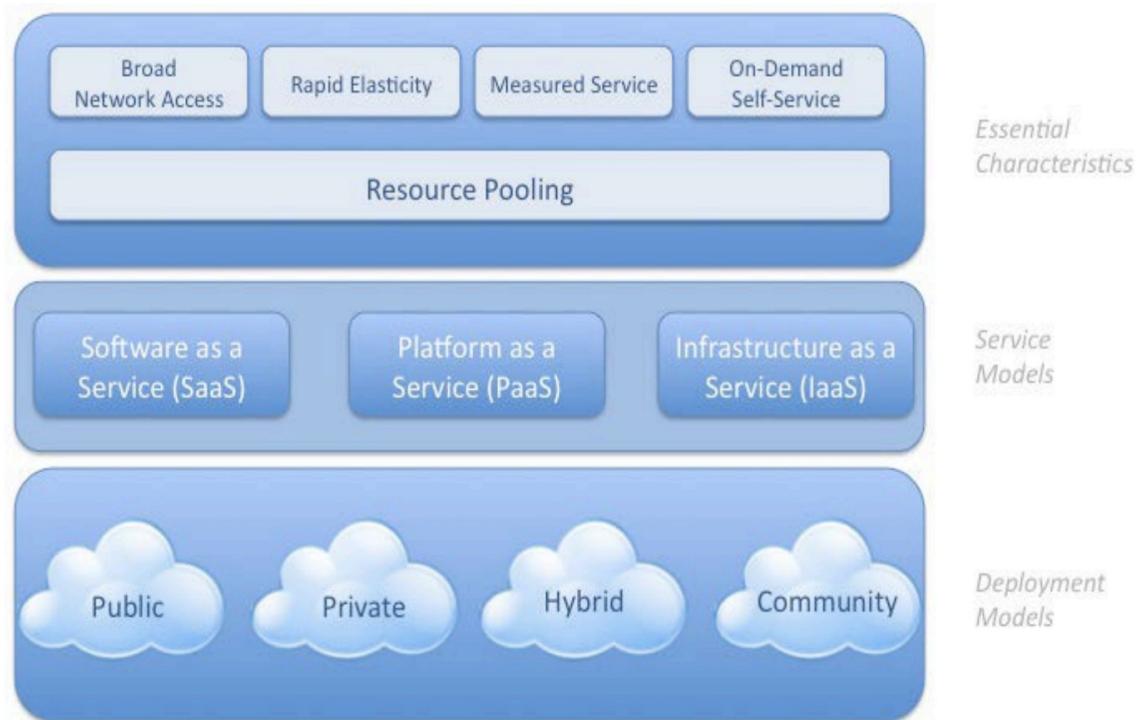


Figure 25: Visual representation of the NIST Cloud Architecture [Source](#)

7.5 CLOUD SECURITY ALLIANCE REFERENCE ARCHITECTURE

Founded in 2008, the Cloud Security Alliance (CSA) is a nonprofit organization that provides a variety of security resources to institutions including guidelines, education and best practices for adoption.

This is a great organization to lean on if you have open questions about architecture and the best way to secure it. There are working groups that look across 38 domains of Cloud Security. These groups meet actively and they cover current topics, opportunities and ask relevant questions. It is a great place to networks with experts in the field and ask questions specific to your company or academic project. You may also find an answer to your question in the white papers, reports, tools, trainings, and services they have available.

The group of industry experts based use the following guiding principles to when publishing their reference Architecture (below).

- Define protections that enable trust in the cloud.
- Develop cross-platform capabilities and patterns for proprietary and open-source providers.
- Will facilitate trusted and efficient access, administration and resiliency to the customer/consumer.
- Provide direction to secure information that is protected by regulations.
- The Architecture must facilitate proper and efficient identification, authentication, authorization, administration and auditability.
- Centralize security policy, maintenance operation and oversight functions.
- Access to information must be secure yet still easy to obtain.
- Delegate or Federate access control where appropriate.
- Must be easy to adopt and consume, supporting the design of security patterns
- The Architecture must be elastic, flexible and resilient supporting multi-tenant, multi-landlord platforms
- The architecture must address and support multiple levels of protection, including network, operating system, and application security needs.

An overview of the architecture is shown in the diagram from the Cloud Security Alliance. See [Figure 26](#)

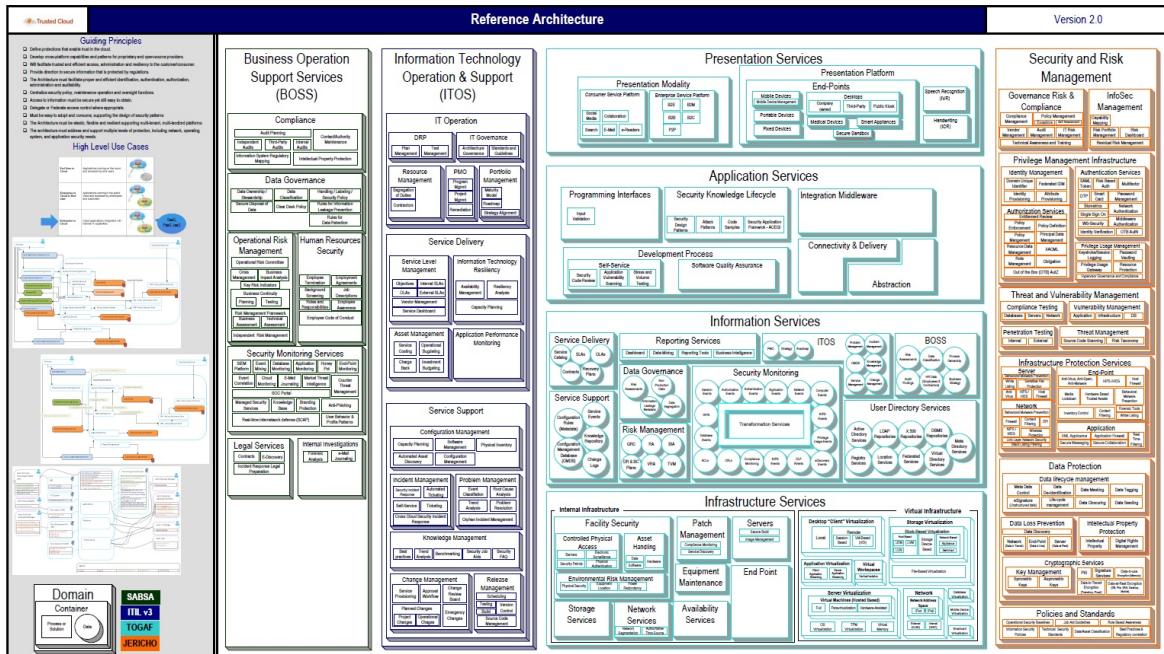


Figure 26: Cloud Security Alliance Reference Architecture
[Source](#)

7.6 MULTICLOUD ARCHITECTURES

One of the issues we see today is that it is unrealistic to assume clouds are only provided by one vendor, or that they have all the same interface. Each vendor is advertising their special services to distinguish themselves from the competitors. For the end user and the developer that projects the problem of vendor lockin. However, we need to be aware of efforts that allow an easy of such vendor lockin while for example providing multi cloud solutions. Such solutions integrate multiple vendors and technologies into a single architecture allowing to use multiple cloud vendors at the same time.

7.6.1 Cloudmesh Architecture

One of the earliest such tools is Cloudmesh.org, which is lead by von Laszewski. The tool was developed at a time when AWS and Nimbus, and Eucalyptus where predominant players. At that time OpenStack was just transitioned from a NASA project to a community development.

FutureGrid ws one of the earliest academic cloud offerings to explore the effectiveness of the different cloud infrastructure solutions. It was clear that a unifying framework and abstraction layer was needed allowing us to utilize the easily. In fact cloudmesh did not only provide a REST based API, but also a commandline shell allowing to switch between clouds with a single variable. It also provided bare metal provisioning before OpenStack even offered it. Through an evolution of developments the current cloudmesh architecture that allows multicloud services is depicted in the next figure. We still distinguish the IaaS level which included not only IaaS Abstractions, but also Containers, and HPC services. Platforms are typically integrated through DevOps that can be hosted on the IaaS. Examples are Hadoop, and Spark The services are exposed through a client API hiding much of the internals to the user. A portal and application services have successfully demonstrated the feasibility of this approach (see [Figure 27](#)).

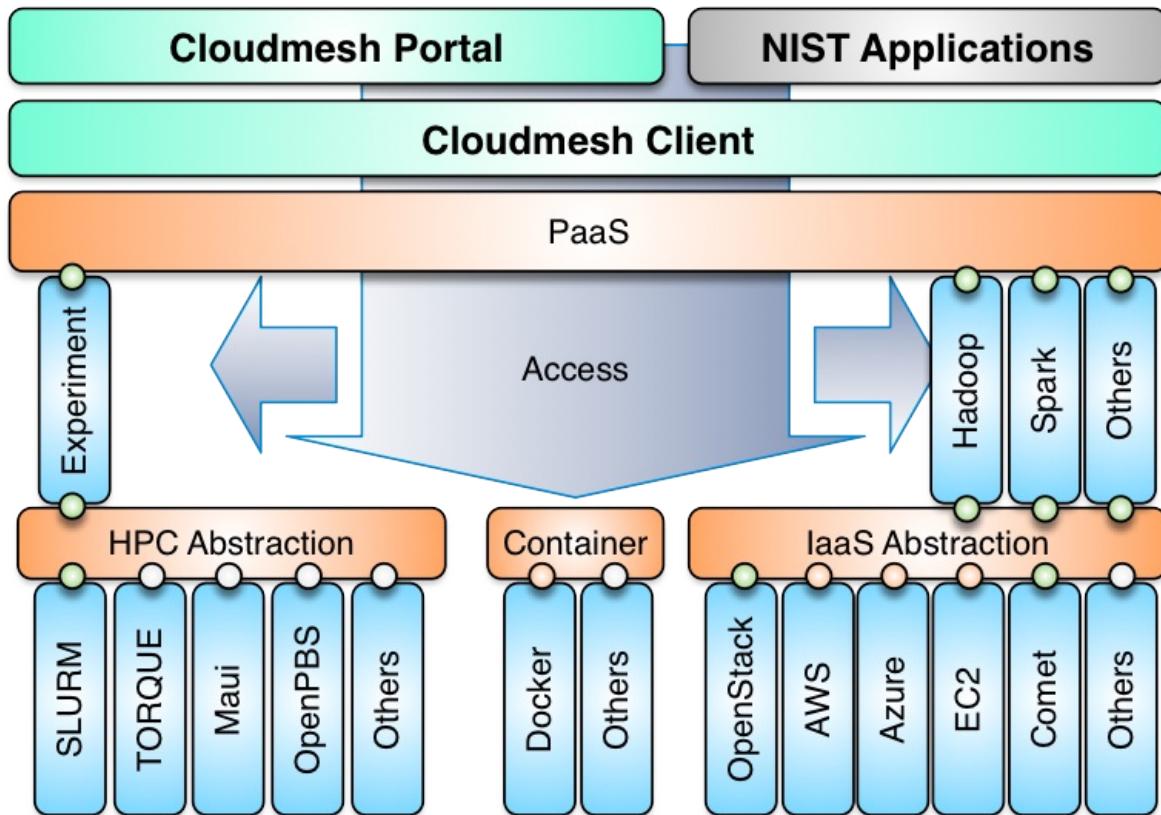


Figure 27: Cloudmesh Arch

Within the hour e516 class we will be developing a modern version of cloudmesh from the ground up by only using python 3 as implementation language, integration of containers, and REST services based on OpenAPI. Local data to manage the different services are hosted in a mongo DB database and exposed through portable containers, so that a single crosplatform environment exists as part of the project deliverables.

Students from e516 can and are in fact expected to participate actively on the development of Cloudmesh v4.0. In addition the OpenAPI service specifications developed for the project will be integrated in Volume 8 of the NIST Big data reference architecture, which is discussed elsewhere.

The advantage of developing such an environment is that we can look at various aspects of cloudcomputing while demonstrating integrated use patterns.

7.7 RESOURCES

- <http://www.lifl.fr/iwaise12/presentations/tata.pdf>
- https://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf
- http://staff.polito.it/alessandro.mantelero/cloud_computing/SiCloud.pdf
- https://resources.sei.cmu.edu/asset_files/Presentation/2010_07.pdf
- <https://www.oracle.com/technetwork/articles/entarch/orgeror-top-10-cloud-1957407.pdf>
- <http://www.oracle.com/technetwork/topics/entarch/oracle-wp-cloud-ref-arch-1883533.pdf>
- <https://pdfs.semanticscholar.org/cecd/c193b73ec1e7b42d1321.pdf>



◆ Learning Objectives

- Obtain an overview of the NIST Big Data Reference Architecture.
- Understand that you can contribute to it as part of this class.

One of the major technical areas in the cloud is to define architectures that can work with Big Data. For this reason NIST has worked now for some time on identifying how to create a data interoperability framework. The idea here is that at one point architecture designers can pick services that they can choose to combine them as part of their data pipeline and integrate in a convenient fashion into their solution.

Besides just being a high level description NIST also encourages the verification of the architecture through interface specifications, especially those that are currently under way in Volume 8 of the document series. You have the unique opportunity to help shape this interface and contribute to it. We will provide you not only mechanisms on how you theoretically can do this, but also how you practically can contribute.

As part of your projects in 516 you will need to integrate a significant service that you can contribute to the NIST document in form of a specification and in form of an implementation.

8.1 PATHWAY TO THE NIST-BDRA

The Nist Big Data Public Working Group (NBD-PWG) was established as collaboration between industry, academia and government "to create a consensus-based extensible Big Data Interoperability Framework (NBDIF) which is a vendor-neutral, technology- and

infrastructure-independent ecosystem” [21]. It will be helpful for Big Data stakeholders such as data architects, data scientists, researchers, implementers to integrate and utilize “the best available analytics tools to process and derive knowledge through the use of standard interfaces between swappable architectural components” [21]. The NBDIF is being developed in three stages:

- Stage 1: “Identify the high-level Big Data reference architecture key components, which are technology, infrastructure, and vendor agnostic,” [21] introduction of the Big Data Reference Architecture (NBD-RA);
- Stage 2: “Define general interfaces between the NBD-RA components with the goals to aggregate low-level interactions into high-level general interfaces and produce set of white papers to demonstrate how NBD-RA can be used” [21];
- Stage 3: “Validate the NBD-RA by building Big Data general applications through the general interfaces.[21]”

Nist has developed the following volumes as listed in Table: BDRA volumes that surround the creation of the NIST-BDRA. We recommend that you take a closer look at these documents as in this section we provide a focussed summary with the aspect of cloud computing in mind.

Table: NIST BDRA Volumes

.

Volumes	Volume	Title
NIST SP1500-1r1	Volume 1	Definitions
NIST SP1500-2r1	Volume 2	Taxonomies
NIST SP1500-3r1	Volume 3	Use Cases and Requirements

<u>NIST SP1500-4r1</u>	Volume 4	Security and Privacy
<u>NIST SP1500-5</u>	Volume 5	Reference Architectures White Paper Survey
<u>NIST SP1500-6r1</u>	Volume 6	Reference Architecture
<u>NIST SP1500-7r1</u>	Volume 7	Standards Roadmap
<u>NIST SP1500-9</u>	Volume 8	Reference Architecture Interface (new)
<u>NIST SP1500-10</u>	Volume 9	Adoption and Modernization (new)

8.2 BIG DATA CHARACTERISTICS AND DEFINITIONS

Volume 1 of the series introduces the community to common definitions that are used as part of the field of Big data. This includes the analysis of characteristics such as volume, velocity, variety, variability and the use of structures and unstructured data. As part of the field of data science and engineering it lists a number of areas that are believed to be essential including that they must master including data structures, parallelism, metadata, flow rate, visual communication. In addition we believe that an additional skill set must be prevalent that allows a data engineer to deploy such technologies onto actual systems.

We have submitted the following proposal to NIST:

3.3.6. Deployments:

A significant challenge exists for data engineers to develop architectures and their deployment implications. The volume of data and the processing power needed to analyze them may require many

thousands of distributed compute resources. They can be part of private data centers, virtualized with the help of virtual machines or containers and even utilize serverless computing to focus integration of Big Data Function as a Service based architectures. As such architectures are assumed to be large community standards such as leveraging DevOps will be necessary for the engineers to setup and manage such architectures. This is especially important with the swift development of the field that may require rolling updates without interruption of the services offered.

This addition reflects the newest insight into what a data scientist needs to know and the newest job trends that we observed.

To identify what big data is we find the following characteristics

Volume: Big for data means lots of bytes. This could be achieved in many different ways. Typically we look at the actual size of a data set, but also how this data set is stored for example in many thousands of smaller files that are part of the data set. It is clear that in many of such cases analysis of a large volume of data will impact the architectural design for storage, but also the workflow on how this data is processed.

Velocity: We see often that big data is associated with high data flow rates caused by for example streaming data. It can however also be caused by functions that are applied to large volumes of data and need to be integrated quickly to return the result as fast as possible. Needs for real time processing as part of the quality of service offered contribute also to this. Examples of IoT devices that integrate not only data in the cloud, but also on the edge need to be considered.

Variety: In today's world we have many different data resources that motivate sophisticated data mashup strategies. Big data hence not only deals with information from one source but a variety of sources. The architectures and services utilized are multiple and needed to enable automated analysis while incorporating various data sources.

Another aspect of variety is that data can be structured or unstructured. NIST finds this aspect so important that they included its own section for it.

Variability: Any data over time will change. Naturally that is not an exception in Big data where data may be a time to live or needs to be updated in order not to be stale or obsolete. Hence one of the characteristics that big data could exhibit is that its data be variable and is prone to changes.

In addition to these general observations we also have to address important characteristics that are attached with the Data itself. This includes

Veracity: Veracity refers to the accuracy of the data. Accuracy can be increased by adding metadata.

Validity: Refers to data that is valid. While data can be accurately measured, it could be invalid by the time it is processed.

Volatility: Volatility refers to the change in the data values over time.

Value: Naturally we can store lots of information, but if the information is not valuable then we may not need to store it. This is recently been seen as a trend as some companies have transitioned data sets to the community as they do not provide value to the service provider to justify its prolonged maintenance. (O Look for an example) In other cases the data has become so valuable and that the services offered have been reduced for example as they provide too many resource needs by the community. A good example is Google scholar that used to have much more liberal use and today its services are significantly scaled back for public users.

8.3 BIG DATA AND THE CLOUD

While looking at the characteristics of Big Data it is obvious that Big data is on the one hand a motivator for cloud computing, but on the

other hand existing Big Data frameworks are a motivator for developing Big Data Architectures a certain way.

Hence we have to always look from both sides towards the creation of architectures related to a particular application of big data.

This is also motivated by the rich history we have in the field of parallel and distributed computing. For a long time engineers have dealt with the issue of horizontal scaling, which is defined by adding more nodes or other resources to a cluster. Such resources may include

- shared disk file systems
- distributed file systems
- distributed data processing and concurrency frameworks, such as Concurrent sequential processes, workflows, MPI, map/reduce, or shared memory
- resource negotiation to establish quality of service
- data movement
- and data tiers (as showcased in high energy physics) ○ add ref to Ligo and Atlas

In addition to the horizontal scaling issues we also have to worry about the vertical scaling issues, this is how the overall system architecture fits together to address an end-to-end use case. In such efforts we look at

- interface designs
- workflows between components and services
- privacy of data and other security issues
- reusability within other use-cases.

Naturally the cloud offers the ability to cloudify existing relational databases as cloud services while leveraging the increased performance and special hardware and software support that may be otherwise unaffordable for an individual user. However we see also the explosive growth of non sql databases because some of them can more effectively deal with the characteristics of big data than

traditional mostly well structured data bases. In addition many of these frameworks are able to introduce advanced capability such as distributed and reliable service integration.

Although we have been used to the term cloud while using virtualized resources and the term Grid by offering a network of supercomputers in a virtual organization, We should not forget that Cloud service providers also offer High performance computers resources for some of their most advanced users. O add reference. Naturally such resources can be used not only for numerical intensive computations but also for big data applications as the Physics community has demonstrated.

8.4 BIG DATA, EDGE COMPUTING AND THE CLOUD

When looking at the number of devices that are being added daily to the global IT infrastructure we observe that cellphones and soon Internet of Things (IoT) devices will produce the bulk of all data. However not all data will be moved to the cloud and lots of data will be analyzed locally on the devices or even not being considered to be uploaded to the cloud either because it project to low or to high value to be moved. However a considerable portion will put new constraints on our services we offer in the cloud and any architecture addressing this must be properly deal with scaling early on in the architectural design process.

8.5 REFERENCE ARCHITECTURE

Next we present the Big data reference architecture. It is Depicted in [Figure 28](#). According to the document (Volume 2) the five main components representing the central roles include

- System Orchestrator: Defines and integrates the required data application activities into an operational vertical system;
- Data Provider: Introduces new data or information feeds into the Big Data system;

- Big Data Application Provider: Executes a life cycle to meet security and privacy requirements as well as System Orchestrator-defined requirements;
- Big Data Framework Provider: Establishes a computing framework in which to execute certain transformation applications while protecting the privacy and integrity of data; and
- Data Consumer: Includes end users or other systems who use the results of the Big Data Application Provider.

In addition we recognize two fabrics layers:

- Security and Privacy Fabric
- Management Fabric

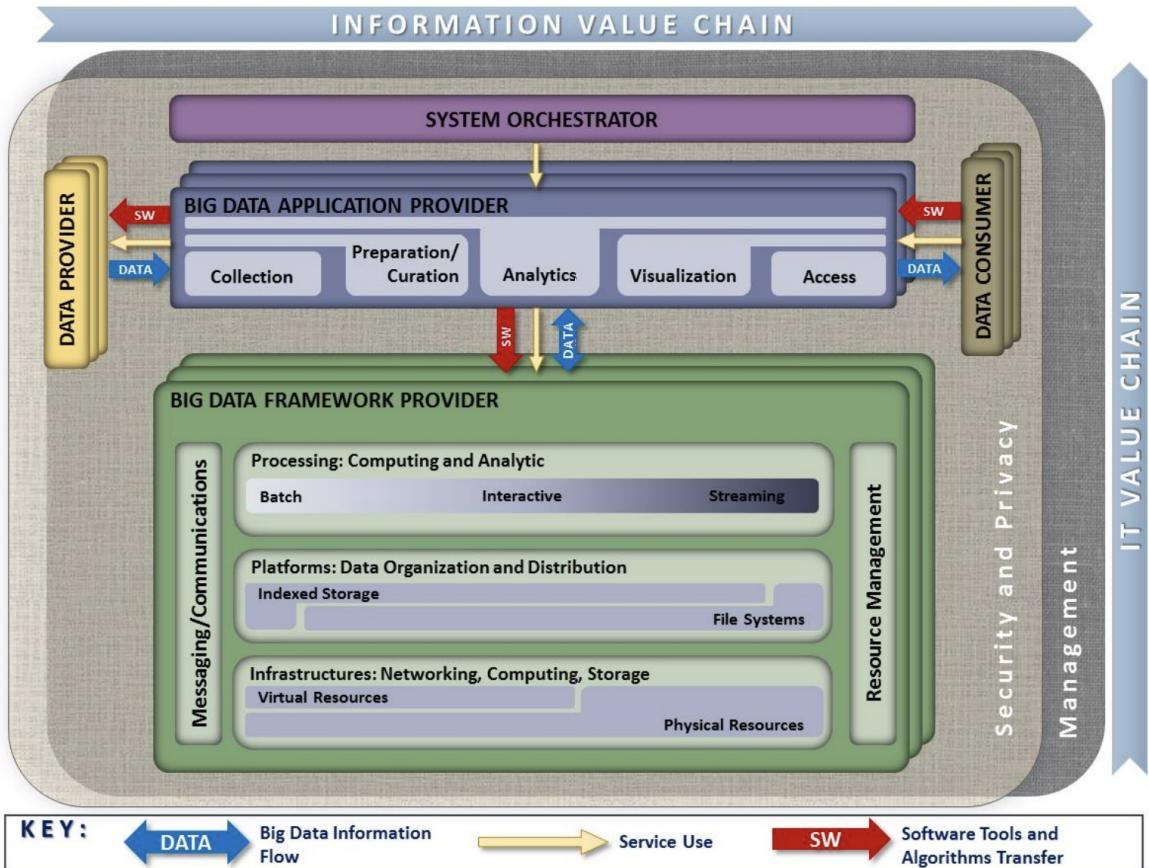


Figure 28: NIST-BDRA (see Volume 2)

While looking at the actors depicted in [Figure 29](#) we need to be aware that in each of the categories a service can be added. This is an important distinction to the original depiction in the definition as it is clear that an automated service could act in behalf of the actors listed in each of the categories.

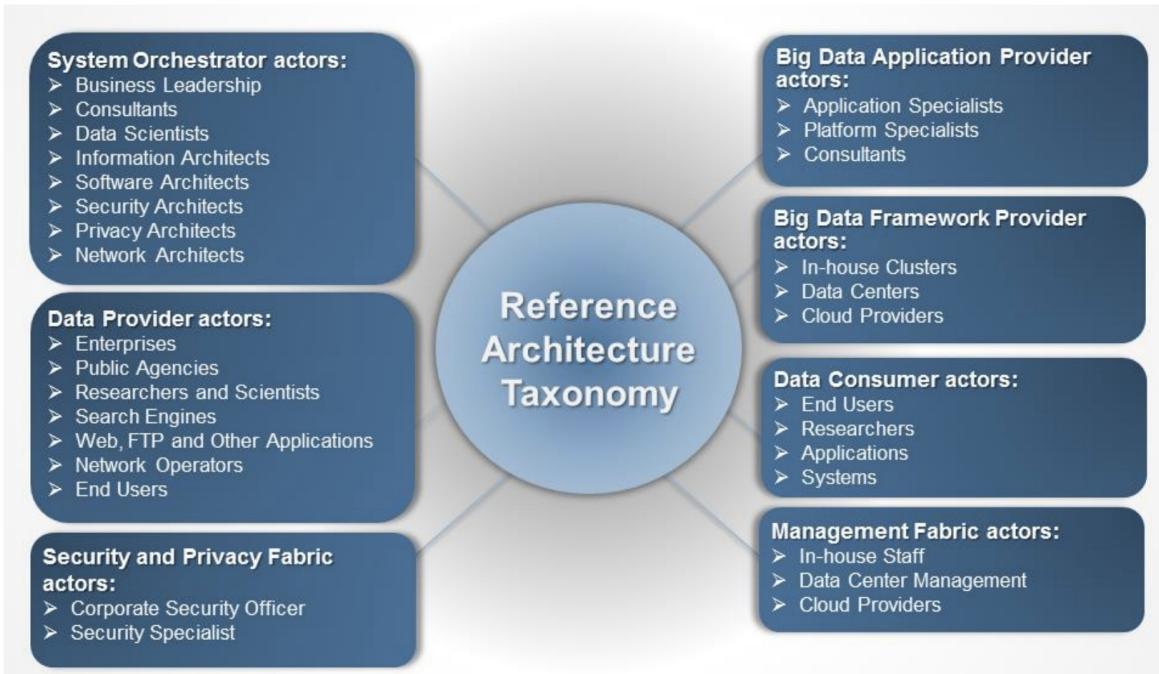


Figure 29: NIST Roles (see Volume 2)

For a detailed definition which is beyond the scope of this document we refer to the Volume 2 of the documents.

8.6 FRAMEWORK PROVIDERS

Traditionally cloud computing has started with offering IaaS, followed by PaaS and SaaS. We see the IaaS reflected in three categories for big data:

1. Traditional compute and network resources including virtualization frameworks
2. Data Organization and Distribution systems such as offered in Indexed Storage and File Systems
3. Processing engines offering batch, interactive, and streaming services to provide computing and analytics activities

Messaging and communication takes place between these layers while resource management is used to address efficiency.

Frameworks such as Spark and Hadoop include components from

multiple of these categories to create a vertical integrated system. Often they are offered by a service provider. However, one needs to be reminded that such offerings may not be tailored to the individual use-case and inefficiencies could be prevalent because the service offer is outdated, or it is not explicitly tuned to the problem at hand.

8.7 APPLICATION PROVIDERS

The underlaying infrastructure is reused by big data application providers supporting services and task such as

- Data collections
- Data curation
- Data Analytics
- Data Visualization
- Data Access

Through the interplay between these services data consumer sand data producers can be served.

8.8 FABRIC

Security and general management are part of the governing fabric in which such an architecture is deployed.

8.9 INTERFACE DEFINITIONS

The interface definitions for the BDRA are specified in Volume 8. We are in the second phase of our document specification while we switch from our pure Resource descripyion to an OpenAPI specification. Before we can provide more details we need to introduce you to REST which is an essential technology for many moder cloud computing services.



🎓 Learning Objectives

- Understand REST Services.
- Understand OpenAPI.
- Develop REST services in Python using Eve.
- Develop REST services in Python using OpenAPI with swagger.

9.1 OVERVIEW OF REST

Test short refid. This should bring up the [python intro](#). This should bring up the [graphql](#).

This section is accompanied by a video about REST.

📹 [REST 36:02](#)

REST stands for **R**epresentational **S**tate **T**ransfer. REST is an architecture style for designing networked applications. It is based on stateless, client-server, cacheable communications protocol. In contrast to what some others write or say, REST is not a standard. Although not based on http, in most cases, the HTTP protocol is used. In that case, RESTful applications use HTTP requests to (a) post data while creating and/or updating it, (b) read data while making queries, and (c) delete data.

REST was first introduced in a thesis from Fielding:

- <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Hence REST can use HTTP for the four CRUD operations:

- Create resources

- **Read** resources
- **Update** resources
- **Delete** resources

As part of the HTTP protocol we have methods such as GET, PUT, POST, and DELETE. These methods can then be used to implement a REST service. This is not surprising as the HTTP protocol was explicitly designed to support these operations. As REST introduces collections and items we need to implement the CRUD functions for them. We distinguish single resources and collection of resources. The semantics for accessing them is explained next illustrating how to implement them with HTTP methods (see https://en.wikipedia.org/wiki/Representational_state_transfer).

9.1.1 Collection of Resources

Let us assume the following URI identifies a collection of resources

`http://.../resources/`

than we need to implement the following CRUD methods:

GET

List the URIs and perhaps other details of the collections members

PUT

Replace the entire collection with another collection.

POST

Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.

DELETE

Delete the entire collection.

9.1.2 Single Resource

Let us assume the following URI identifies a single resource in a collection of resources

`http://.../resources/item42`

than we need to implement the following CRUD methods:

GET

Retrieve a representation of the addressed member of the collection, expressed in an appropriate internet media type.

PUT

Replace the addressed member of the collection, or if it does not exist, create it.

POST

Not generally used. Treat the addressed member as a collection in its own right and create a new entry within it.

DELETE

Delete the addressed member of the collection.

9.1.3 REST Tool Classification

Due to the well defined structure that REST provides a number of tools have been created that manage the creation of the specification for rest services and their programming. We distinguish several different categories:

REST programming language support:

These tools and services are targeting a particular programming language. Such tools include Eve which we will explore in more detail.

REST documentation based tools:

These tools are primarily focusing on documenting REST specifications. Such tools include Swagger, which we will explore in more detail.

REST design support tools:

These tools are used to support the design process of developing REST services while abstracting on top of the programming languages and define reusable specifications that can be used to create clients and servers for particular technology targets. Such tools include also swagger as additional tools are available that can generate code from swagger specifications, which we will explore in more detail.

9.2 OPENAPI REST SERVICES WITH SWAGGER



Swagger <https://swagger.io/> is a tool for developing API specifications based on the OpenAPI Specification (OAS). It allows not only the specification, but the generation of code based on the specification in a variety of languages.

Swagger itself has a number of tools which together build a framework for developing REST services for a variety of languages.

9.2.1 Swagger Tools

The major Swagger tools of interest are:

Swagger Core

includes libraries for working with Swagger specifications
<https://github.com/swagger-api/swagger-core>.

Swagger Codegen

allows to generate code from the specifications to develop Client SDKs, servers, and documentation. <https://github.com/swagger-api/swagger-codegen>

Swagger UI

is an HTML5 based UI for exploring and interacting with the specified APIs <https://github.com/swagger-api/swagger-ui>

Swagger Editor

is a Web-browser based editor for composing specifications using YAML <https://github.com/swagger-api/swagger-editor>

The developed APIs can be hosted and further developed on an online repository named SwaggerHub <https://app.swaggerhub.com/home> The convenient online editor is available which also can be installed locally on a variety of operating systems including macOS, Linux, and Windows.

9.2.2 Swagger Community Tools

notify us about other tools that you find and would like us to mention here.

9.2.2.1 Swagger Toolbox

Swagger toolbox is a utility that can convert json to swagger compatible yaml models. It is hosted online at

- <https://swagger-toolbox.firebaseio.com/>

The source code to this tool is available on github at

.

It is important to make sure that the json model is properly configured. As such each datatype must be wrapped in “quotes” and the last element must not have a `,` behind it.

In case you have large models, we recommend that you gradually add more and more features so that it is easier to debug in case of an error. This tool is not designed to provide back a full featured OpenAPI, but help you getting started deriving one.

Let us look at a small example. Let us assume we want to create a REST service to execute a command on the remote service. We know this may not be a good idea if it is not properly secured, so be extra careful. A good way to simulate this is to just use a return string instead of executing the command.

Let us assume the json schema looks like:

```
{  
    "host": "string",  
    "command": "string"  
}
```

The output the swagger toolbox creates is

```
---  
required:  
- "host"  
- "command"  
properties:  
  host:  
    type: "string"  
  command:  
    type: "string"
```

As you can see it is far from complete, but it could be used to get you started.

Based on this tool develop a rest service to which you send a schema in JSON format from which you get back the YAML model.

9.3 OPENAPI SPECIFICATION



Swagger provides through its specification the definition of REST

services through a YAML or JSON document.

When following the API-specification-first approach to define and develop a RESTful service, the first and foremost step is to define the API conforming to the OpenAPI specification, and then using codegen tools to conveniently generate server side stub code, client code, documentations, in the language you desire. In Section [REST Service Generation with OpenAPI](#) we have introduced the codegen tool and how to use that to generate server side and client side code and documentation. In this Section [The Virtual Cluster example API Definition](#) we will use a slightly more complex example to show how to define an API following the OpenAPI 2.0 specification. The example is to retrieve virtual cluster (VC) object from the server.

The OpenAPI Specification is formerly known as Swagger RESTful API Documentation Specification. It defines a specification to describe and document a RESTful service API. It is also known under version 3.0 of swagger. However, as the tools for 3.0 are not yet completed, we will continue for now to use version swagger 2.0, till the transition has been completed. This is especially of importance, as we need to use the swagger codegen tool, which currently support only up to specification v2. Hence we are at this time using OpenAPI/Swagger v2.0 in our example. There are some structure and syntax changes in v3, while the essence is very similar. For more details of the changes between v3 and v2, please refer to A document published on the Web titled [Difference between OpenAPI 3.0 and Swagger 2.0](#).

You can write the API definition in json for yaml format. Let us discuss this format briefly and focus on yaml as it is easier to read and maintain.

On the root level of the yaml document we see fields like swagger, info, and so on. Among these fields, **swagger**, **info**, and **path** are **required**. Their meaning is as follows:

swagger

specifies the version number. IN our case a string value '2.0' is

used as we are writing the definition conforming to the v2.0 specification.

info

defines metadata information related to the API. E.g., the API version, title and description, termsOfService if applicable, contact information and license, etc. Among these attributes, **version** and **title** are required while others are optional.

path

defines the actual endpoints of the exposed RESTful API service. Each endpoint has a field pattern as the key, and a Path Item Object as the value. In this example we have defined /vc and /vc/{id} as the two service endpoints. They will be part of the final service URL, appended after the service host and basePath, which will be explained later.

Let us focus on the Path Item Object. It contains one or more supported operations on the service endpoint. An operation is keyed by a valid HTTP operation verb, e.g., one of **get**, **put**, **post**, **delete**, or **patch**. It has a value of Operation Object that describes the operations in more detail.

The Operation Object will always **require** a Response Object. A Response Object has a HTTP status code as the key, e.g., **200** as successful return; **40X** as authentication and authorization related errors; and **50x** as other server side servers. It can also has a default response keyed by **default** for undeclared http status return code. The Response Object value has a **required** description field, and if anything is returned, a schema indicating the object type to be returned, which could be a primitive type, e.g., string, or an array or customized object. In case of object or an array of object, use \$ref to point to the definition of the object. In this example, we have

\$ref: "#/definitions/VC"

to point to the VC definition in the definitions section in the same specification file, which will be explained later.

Besides the required field, the Operation Object **can** have summary and description to indicate what the operation is about; and operationId to uniquely identify the operation; and consumes and produces to indicate what MIME types it expects as input and for returns, e.g., application/json in most modern RESTful APIs. It can further specify what input parameter is expected using parameters, which requires a name and in fields. name specifies the name of the parameter, and in specifies from where to get the parameter, and its possible values are query, header, path, formData or body. In this example in the /vc/{id} path we obtain the id parameter from the URL path so it has the path value. When the in has path as its value, the required field is required and has to be set as true; when the in has value other than body, a type field is required to specify the type of the parameter.

While the three root level fields mentioned previously are required, in most cases we will also use other optional fields.

host

to indicate where the service is to be deployed, which could be localhost or a valid IP address or a DNS name of the host where the service is to be deployed. If other port number other than 80 is to be used, write the port number as well, e.g., localhost:8080.

schemas

to specify the transfer protocol, e.g, http or https.

basePath

to specify the common base URL to be append after the host to form the base path for all the endpoints, e.g., /api or /api/1.0/. In this example with the values specified we would have the final

service endpoints `http://localhost:8080/api/vcs` and `http://localhost:8080/api/vc/{id}` by combining the schemas, host, basePath and paths values.

consumes and produces

can also be specified on the top level to specify the default MIME types of the input and return if most paths and the defined operations have the same.

definitions

as used in in the paths field, in order to point to a customized object definition with a \$ref keyword.

The definitions field really contains the object definition of the customized objects involved in the API, similar to a class definition in any Object Oriented programming language. In this example, we defined a VC object, and hierarchically a Node object. Each object defined is a type of Schema Object in which many field could be used to specify the object (see details in the REF link at top of the document), but the most frequently used ones are:

type

to specify the type, and in the customized definition case the value is mostly object.

required

field to list the names of the required attributes of the object.

properties

has the detailed information of each attribute/property of the object, e.g, type, format. It also supports hierarchical object definition so a property of one object could be another customized object defined elsewhere while using schema and

\$ref keyword to point to the definition. In this example we have defined a VC, or virtual cluster, object, while it contains another object definition of

Node

as part of a cluster.

9.3.1 The Virtual Cluster example API Definition

9.3.1.1 Terminology

VC

A virtual cluster, which has one Front-End (FE) management node and multiple compute nodes. A VC object also has id and name to identify the VC, and nnodes to indicate how many compute nodes it has.

FE

A management node from which to access the compute nodes. The FE node usually connects to all the compute nodes via private network.

Node

A computer node object that the info ncores to indicate number of cores it has, and ram and localdisk to show the size of RAM and local disk storage.

9.3.1.2 Specification

```
swagger: "2.0"
info:
  version: "1.0.0"
  title: "A Virtual Cluster"
  description: "Virtual Cluster as a test of using swagger-2.0 specification and codegen"
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "IU ISE software and system team"
```

```

license:
  name: "Apache"
host: "localhost:8080"
basePath: "/api"
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:
  /vcs:
    get:
      description: "Returns all VCs from the system that the user has access to"
      produces:
        - "application/json"
      responses:
        "200":
          description: "A list of VCs."
          schema:
            type: "array"
            items:
              $ref: "#/definitions/VC"
  /vcs/{id}:
    get:
      description: "Returns all VCs from the system that the user has access to"
      operationId: getVCById
      parameters:
        - name: id
          in: path
          description: ID of VC to fetch
          required: true
          type: string
      produces:
        - "application/json"
      responses:
        "200":
          description: "The vc with the given id."
          schema:
            $ref: "#/definitions/VC"
      default:
        description: unexpected error
        schema:
          $ref: '#/definitions/Error'
definitions:
  VC:
    type: "object"
    required:
      - "id"
      - "name"
      - "nnodes"
      - "FE"
      - "computes"
    properties:
      id:
        type: "string"
      name:
        type: "string"
      nnodes:
        type: "integer"

```

```

    format: "int64"
FE:
  type: "object"
  schema:
    $ref: "#/definitions/Node"
computes:
  type: "array"
  items:
    $ref: "#/definitions/Node"
tag:
  type: "string"
Node:
  type: "object"
  required:
    - "ncores"
    - "ram"
    - "localdisk"
  properties:
    ncores:
      type: "integer"
      format: "int64"
    ram:
      type: "integer"
      format: "int64"
    localdisk:
      type: "integer"
      format: "int64"
Error:
  required:
    - code
    - message
  properties:
    code:
      type: integer
      format: int32
    message:
      type: string

```

9.3.2 References

[The official OpenAPI 2.0 Documentation](#)

9.4 OPENAPI REST SERVICE VIA INTROSPECTION



The simplest way to create an OpenAPI service is to use the connexion service and read in the specification from its yaml file. It will then be introspected and dynamically methods are created that are used for the implementation of the server.

The full example for this is available in

- <https://github.com/cloudmesh-community/nist/tree/master/examples/flask-connexion-swagger>

This example will return dynamically the cpu information of a computer.

Our requirements.txt file includes

```
flask
connexion
```

as dependencies. The `server.py` file simply contains the following code:

```
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("cpu.yaml")

# create a URL route in our application for "/"
@app.route("/")
def home():
    msg = {"msg": "It's working!"}
    return jsonify(msg)

if __name__ == "__main__":
    app.run(port=8080, debug=True)
```

This will run our REST service under the assumption we have a `cpu.yaml` and a `cpu.py` files as our yaml file calls out methods from `cpu.py`

The yaml file looks as follows

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "cpuinfo"
  description: "A simple service to get cpufreq as an example of using swagger-2.0 specific"
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "Cloudmesh REST Service Example"
  license:
    name: "Apache"
host: "localhost:8080"
basePath: "/cloudmesh"
```

```

schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:
  /cpu:
    get:
      tags:
        - CPU
      operationId: cpu.get_processor_name
      description: "Returns cpu information of the hosting server"
      produces:
        - "application/json"
      responses:
        "200":
          description: "CPU info"
          schema:
            $ref: "#/definitions/CPU"
definitions:
  CPU:
    type: "object"
    required:
      - "model"
    properties:
      model:
        type: "string"

```



Here we simply implement a get method and associate it with the URL /cpu. The operationid, defines the method that we call which as we used the local directory is included in the file `cpu.py`. This is controlled by the prefix in the operation id.

A very simple function to return the cpu information is defined in `cpu.py` which we list next

```

import os, platform, subprocess, re
from flask import jsonify

def get_processor_name():
    if platform.system() == "Windows":
        p = platform.processor()
    elif platform.system() == "Darwin":
        command = "/usr/sbin/sysctl -n machdep.cpu.brand_string"
        p = subprocess.check_output(command, shell=True).strip().decode()
    elif platform.system() == "Linux":
        command = "cat /proc/cpuinfo"
        all_info = subprocess.check_output(command, shell=True).strip().decode()
        for line in all_info.split("\n"):
            if "model name" in line:
                p = re.sub(".*model name.*:", "", line, 1)
    else:
        p = "cannot find cpuinfo"

```

```
pinfo = {"model": p}
return jsonify(pinfo)
```

We have implemented this function to return a jsonified information from the dict pinfo.

To simplify working with this example, we also provide a makefile for OSX that allows us to call the server and the call to the servoer in two different terminals

```
define terminal
    osascript -e 'tell application "Terminal" to do script "cd $(PWD); $1"'
endef

install:
    pip install -r requirements.txt

demo:
    $(call terminal, python server.py)
    sleep 3
    @echo =====
    @echo "Get the info"
    @echo =====
    curl http://localhost:8080/cloudmesh/cpu
    @echo
    @echo =====
```

When we call

```
make demo
```

our demo is run.

9.4.1 Exercise

OpenAPI.Conexion.1:

Modify the makefile so it works also on ubuntu, but do not disable the ability to run it correctly on OSX. Tip use if's in makefiles base on the OS. You can look at the makefiles that create this book as example. find alternatives to sarting a terminal in Linux.

OpenAPI.Conexion.2:

Modify the makefile so it works also on Windows 10, but do not disable the ability to run it correctly on OSX. Tip use ifs in makefiles. You can look at the makefiles that create this book as example. Find alternatives to start a powershell or cmd.exe in windows. Maybe you need to use gitbash.

OpenAPI.Conexion.3:

Implement a swagger specification of an issue related to the NIST BDRA. Implement it. Please remember this could prepare you for a project good topics include:

- virtual compute service interfacing with aws, azure, google or openstack
- virtual directory service interfacing with google drive, box, github, iCloud, ftp, scp, and others

As there are so many possibilities to contribute, come up in class with one specification and than implement it for different providers. The difficulty here is that it is not done for one IaaS, but for all of them and all can be integrated.

This exercise is typically growing to be part of your class project.

9.5 OPENAPI REST SERVICE VIA CODEGEN



REST 36:02 Swagger

In this subsection we are discussing how to use OpenAPI 2.0 and Swagger Codegen to define and develop a REST Service.

We assume you have been familiar with the concept of REST service, OpenAPI as discussed in section [Overview of Rest](#).

In next section we will further look into the Swagger/OpenAPI 2.0 specification [Swagger Specification](#) and use a slight more complex example to walk you through the design of a RESTful service following

the OpenAPI 2.0 specifications.

We will use a simple example to demonstrate the process of developing a REST service with Swagger/OpenAPI 2.0 specification and the tools related to it. The general steps are:

- Step 1 (Section [Step 1: Define Your REST Service](#)). Define the REST service conforming to Swagger/OpenAPI 2.0 specification. It is a YAML document file with the basics of the REST service defined, e.g., what resources it has and what actions are supported.
- Step 2 (Section [Step 2: Server Side Stub Code Generation and Implementation](#)). Use Swagger Codegen to generate the server side stub code. Fill in the actual implementation of the business logic portion in the code.
- Step 3 (Section [Step 3: Install and Run the REST Service](#)). Install the server side code and run it. The service will then be available.
- Step 4 (Section [Step 4: Generate Client Side Code and Verify](#)). Generate client side code. Develop code to call the REST service. Install and run to verify.

9.5.1 Step 1: Define Your REST Service

In this example we define a simple REST service that returns the hosting server's basic CPU info. The example specification in yaml is as follows:

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "cpuinfo"
  description: "A simple service to get cpuinfo as an example of using swagger-2.0 specific"
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "Cloudmesh REST Service Example"
  license:
    name: "Apache"
host: "localhost:8080"
```

```
basePath: "/api"
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:
  /cpu:
    get:
      description: "Returns cpu information of the hosting server"
      produces:
        - "application/json"
      responses:
        "200":
          description: "CPU info"
          schema:
            $ref: "#/definitions/CPU"
definitions:
  CPU:
    type: "object"
    required:
      - "model"
    properties:
      model:
        type: "string"
```

9.5.2 Step 2: Server Side Stub Code Generation and Implementation

With the REST service having been defined, we can now generate the server side stub code easily.

9.5.2.1 Setup the Codegen Environment

You will need to [install the Swagger Codegen tool](#) if not yet done so. For macOS we recommend that you use the homebrew install via

```
$ brew install swagger-codegen
```

On Ubuntu you can install swagger as follows (update the version as needed):

```
$ mkdir ~/swagger
$ cd ~/swagger
$ wget https://oss.sonatype.org/content/repositories/releases/io/swagger/swagger-codegen-cli-2.3.1.jar
$ alias swagger-codegen="java -jar ~/swagger/swagger-codegen-cli-2.3.1.jar"
```

Add the alias to your `.bashrc` or `.bash_profile` file. After you start a new terminal you can use in that terminal now the command

```
swagger-codegen
```

For other platforms you can just use the `.jar` file, which can be downloaded from [this link](#).

Also make sure Java 7 or 8 is installed in your system. To have a well defined location we recommend that you place the code in the directory `~/cloudmesh`. In this directory you will also place the `cpu.yaml` file.

9.5.2.2 Generate Server Stub Code

After you have the codegen tool ready, and with Java 7 or 8 installed in your system, you can run the following to generate the server side stub code:

```
$ swagger-codegen generate \
-i ~/cloudmesh/cpu.yaml \
-l python-flask \
-o ~/cloudmesh/swagger_example/server/cpu/flaskConnexion \
-D supportPython2=true
```

or if you have not created an alias

```
$ java -jar swagger-codegen-cli.jar generate \
-i ~/cloudmesh/cpu.yaml \
-l python-flask \
-o ~/cloudmesh/swagger_example/server/cpu/flaskConnexion \
-D supportPython2=true
```

In the specified directory under `flaskConnexion` you will find the generated python flask code, with python 2 compatibility. It is best to place the swagger code under the directory `~/cloudmesh` to have a location where you can easily find it. If you want to use python 3 make sure to chose the appropriate option. To switch between python 2 and python 3 we recommend that you use pyenv as discussed in our python section.

9.5.2.3 Fill in the actual implementation

Under the flaskConnexion directory, you will find a swagger_server directory, under which you will find directories with models defined and controllers code stub resides. The models code are generated from the definition in Step 1. On the controller code though, we will need to fill in the actual implementation. You may see a `default_controller.py` file under the controllers directory in which the resource and action is defined but yet to be implemented. In our example, you will find such a function definition which we list next:

```
def cpu_get(): # noqa: E501
    """cpu_get

    Returns cpu info of the hosting server # noqa: E501

    :rtype: CPU
    """
    return 'do some magic!'
```

Please note the `do some magic!` string at the return of the function. This ought to be replaced with actual implementation what you would like your REST call to be really doing. In reality this could be some call to a backend database or datastore; a call to another API; or even calling another REST service from another location. In this example we simply retrieve the cpu model information from the hosting server through a simple python call to illustrate this principle. Thus you can define the following code:

```
import os, platform, subprocess, re

def get_processor_name():
    if platform.system() == "Windows":
        return platform.processor()
    elif platform.system() == "Darwin":
        command = "/usr/sbin/sysctl -n machdep.cpu.brand_string"
        return subprocess.check_output(command, shell=True).strip()
    elif platform.system() == "Linux":
        command = "cat /proc/cpuinfo"
        all_info = subprocess.check_output(command, shell=True).strip()
        for line in all_info.split("\n"):
            if "model name" in line:
                return re.sub(".*model name.*:", "", line, 1)
    return "cannot find cpuinfo"
```

And then change the `cpu_get()` function to the following:

```
def cpu_get(): # noqa: E501
```

```
"""cpu_get

Returns cpu info of the hosting server # noqa: E501

:rtype: CPU
"""
return CPU(get_processor_name())
```

Please note we are returning a CPU object as defined in the API and later generated by the codegen tool in the models directory.

It is best not to include the definition of `get_processor_name()` in the same file as you see the definition of `cpu_get()`. The reason for this is that in case you need to regenerate the code, your modified code will naturally be overwritten. Thus, to minimize the changes, we do recommend to maintain that portion in a different filename and import the function as to keep the modifications small.

At this step we have completed the server side code development.

9.5.3 Step 3: Install and Run the REST Service:

Now we can install and run the REST service. It is strongly recommended that you run this in a pyenv or a virtualenv environment.

9.5.3.1 Start a virtualenv:

In case you are not using pyenv, please use virtual env as follows:

```
$ virtualenv RESTServer
$ source RESTServer/bin/activate
```

9.5.3.2 Make sure you have the latest pip:

```
$ pip install -U pip
```

9.5.3.3 Install the requirements of the server side code:

```
$ cd ~/cloudmesh/swagger_example/server/cpu/flaskConnexion
$ pip install -r requirements.txt
```

9.5.3.4 Install the server side code package:

Under the same directory, run:

```
$ python setup.py install
```

9.5.3.5 Run the service

Under the same directory:

```
$ python -m swagger_server
```

You should see a message like this:

```
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

9.5.3.6 Verify the service using a web browser:

Open a web browser and visit:

- <http://localhost:8080/api/cpu>

to see if it returns a json object with cpu model info in it.

Assignment: How would you verify that your service works with a `curl` call?

9.5.4 Step 4: Generate Client Side Code and Verify

In addition to the server side code swagger can also create a client side code.

9.5.4.1 Client side code generation:

Generate the client side code in a similar fashion as we did for the server side code:

```
$ java -jar swagger-codegen-cli.jar generate \
    -i ~/cloudmesh/cpu.yaml \
    -l python \
```

```
-o ~/cloudmesh/swagger_example/client/cpu \
-D supportPython2=true
```

9.5.4.2 Install the client side code package:

Although we could have installed the client in the same python pyenv or virtualenv, we showcase here that it can be installed in a completely different environment. That would make it even possible to use a python 3 based client and a python 2 based server showcasing interoperability between python versions (although we just use python 2 here). Thus we create a new python virtual environment and conduct our install.

```
$ virtualenv RESTClient
$ source RESTClient/bin/activate
$ pip install -U pip
$ cd swagger_example/client/cpu
$ pip install -r requirements.txt
$ python setup.py install
```

9.5.4.3 Using the client API to interact with the REST service

Under the directory `swagger_example/client/cpu` you will find a `README.md` file which serves as an API documentation with example client code in it. E.g., if we save the following code into a `.py` file:

```
from __future__ import print_function
import time
import swagger_client
from swagger_client.rest import ApiException
from pprint import pprint
# create an instance of the API class
api_instance = swagger_client.DefaultApi()

try:
    api_response = api_instance.cpu_get()
    pprint(api_response)
except ApiException as e:
    print("Exception when calling DefaultApi->cpu_get: %s\n" % e)
```

We can then run this code to verify the calling to the REST service actually works. We are expecting to see a return similar to this:

```
{'model': 'Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz'}
```

Obviously, we could have applied additional cleanup of the information returned by the python code, such as removing duplicated spaces.

9.5.5 Towards a Distributed Client Server

Although we develop and run the example on one localhost machine, you can separate the process into two separate machines. E.g., on a server with external IP or even DNS name to deploy the server side code, and on a local laptop or workstation to deploy the client side code. In this case please make changes on the API definition accordingly, e.g., the **host** value.

9.5.6 Exercises

- Links are not yet integrated

E.OpenAPI.1:

In Section [Step 1: Define Your REST Service](#), we introduced a schema. The question relates to termsOfService: Investigate what the termOfService attribute is and suggest a better value. Discuss on piazza.

E.OpenAPI.2:

In Section [Step 1: Define Your REST Service](#), we introduced a schema. The question relates to model: What is the meaning of model under the definitions?

E.OpenAPI.3:

In Section [Step 1: Define Your REST Service](#), we introduced a schema. The question relates to \$ref: what is the meaning of the \$ref. Discuss on piazza, come up with a student answer in class.

E.OpenAPI.1:

In Section [Step 1: Define Your REST Service](#), we introduced a schema. What does the response 200 mean. Do you need other responses?

After you have gone through the entire section and verified it works for you add create a more sophisticated schema and add more attributes exposing more information from your system.

How can you for example develop a rest service that exposes portions of your file system serving large files, e.g. their filenames and their size? How would you download these files? Would you use a rest service, or would you register an alternative service such as ftp, DAV, or others? Please discuss in piazza. Note this will be a helping you to prepare a larger assignment. Think about this first before you implement.

You can try expand the API definition with more resources and actions included. E.g., to include more detailed attributes in the CPU object and to have those information provided in the actual implementation as well. Or you could try defining totally different resources.

The codegen tool provides a convenient way to have the code stubs ready, which frees the developers to focus more on the API definition and the real implementation of the business logic. Try with complex implementation on the back end server side code to interact with a database/datastore or a 3rd party REST service.

For advanced python users, you can naturally use function assignments to replace the `cpu_get()` entirely even after loading the instantiation of the server.

However, this is not needed. If you are an advanced python developer, please feel free to experiment and let us know how you suggest to integrate things easily.

9.6 FLASK RESTFUL SERVICES



Flask is a micro services framework allowing to write web services in python quickly. One of its extensions is Flask-RESTful. It adds for building REST APIs based on a class definition making it relatively simple. Through this interface we can then integrate with your existing Object Relational Models and libraries. As Flask-RESTful leverages the main features from Flask an extensive set of documentation is available allowing you to get started quickly and thoroughly. The Web page contains extensive documentation:

- <https://flask-restful.readthedocs.io/en/latest/>

We will provide a simple example that showcases some hard coded data to be served as a rest service. It will be easy to replace this for example with functions and methods that obtain such information dynamically from the operating system.

This example has not been tested., We like that the class defines a beautiful example to contribute to this section. and explains what happens in this example.

```
from flask import Flask
from flask_restful import reqparse, abort
from flask_restful import Api, Resource

app = Flask(__name__)
api = Api(app)

COMPUTERS = {
    'computer1': {
        'processor': 'iCore7'
    },
    'computer2': {
        'processor': 'iCore5'
    },
    'computer3': {
        'processor': 'iCore3'
    },
}
```

```

def abort_if_cluster_doesnt_exist(computer_id):
    if computer_id not in COMPUTERS:
        abort(404, message="Computer {} does not exist".format(computer_id))

parser = reqparse.RequestParser()
parser.add_argument('processor')

class Computer(Resource):
    ''' shows a single computer item and lets you delete a computer
        item.'''
    def get(self, computer_id):
        abort_if_computer_doesnt_exist(computer_id)
        return COMPUTERS[computer_id]

    def delete(self, computer_id):
        abort_if_computer_doesnt_exist(computer_id)
        del COMPUTERS[computer_id]
        return '', 204

    def put(self, computer_id):
        args = parser.parse_args()
        processor = {'processor': args['processor']}
        COMPUTERS[computer_id] = processor
        return processor, 201

# ComputerList
class ComputerList(Resource):
    ''' shows a list of all computers, and lets you POST to add new computers'''

    def get(self):
        return COMPUTERS

    def post(self):
        args = parser.parse_args()
        computer_id = int(max(COMPUTERS.keys()).lstrip('computer')) + 1
        computer_id = 'computer%i' % computer_id
        COMPUTERS[computer_id] = {'processor': args['processor']}
        return COMPUTERS[computer_id], 201

    ##
    ## Setup the Api resource routing here
    ##
    api.add_resource(ComputerList, '/computers')
    api.add_resource(Computer, '/computers/<computer_id>')

if __name__ == '__main__':
    app.run(debug=True)

```

9.7 REST SERVICES WITH EVE



Next, we will focus on how to make a RESTful web service with Python Eve. Eve makes the creation of a REST implementation in python easy. More information about Eve can be found at:

- <http://python-eve.org/>

Although we do recommend Ubuntu 17.04, at this time there is a bug that forces us to use 16.04. Furthermore, we require you to follow the instructions on how to install pyenv and use it to set up your python environment. We recommend that you use either python 2.7.14 or 3.6.4. We do not recommend you to use anaconda as it is not suited for cloud computing but targets desktop computing. If you use pyenv you also avoid the issue of interfering with your system wide python install. We do recommend pyenv regardless if you use a virtual machine or are working directly on your operating system. After you have set up a proper python environment, make sure you have the newest version of pip installed with

```
$ pip install pip -U
```

To install Eve, you can say

```
$ pip install eve
```

As Eve also needs a backend database, and as MongoDB is an obvious choice for this, we will have to first install MongoDB. MongoDB is a Non-SQL database which helps to store light weight data easily.

9.7.1 Ubuntu install of MongoDB

On Ubuntu you can install MongoDB as follows

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 \
  --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu \
  xenial/mongodb-org/3.6 multiverse" | \
  sudo tee /etc/apt/sources.list.d/mongodb-org-3.6.list
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
```

9.7.2 macOS install of MongoDB

On macOS you can use the command

```
$ brew update  
$ brew install mongodb
```

9.7.3 Windows 10 Installation of MongoDB

A student or student group of this class are invited to discuss on piazza on how to install mongoDB on Windows 10 and come up with an easy installation solution. Naturally we have the same 2 different ways on how to run mongo. In user space or in the system. As we want to make sure your computer stays secure. the solution must have an easy way on how to shut down the Mongo services.

An enhancement of this task would be to integrate this function into cloudmesh cmd5 with a command mongo that allows for easily starting and stopping the service from cms.

9.7.4 Database Location

After downloading Mongo, create the db directory. This is where the Mongo data files will live. You can create the directory in the default location and assure it has the right permissions. Make sure that the /data/db directory has the right permissions by running

9.7.5 Verification

In order to check the MongoDB installation, please run the following commands in one terminal:

```
$ mkdir -p ~/cloudmesh/data/db  
$ mongod --dbpath ~/cloudmesh/data/db
```

In another terminal we try to connect to mongo and issue a mongo command to show the databases:

```
$ mongo --host 127.0.0.1:27017  
$ show databases
```

If they execute without errors, you have successfully installed MongoDB. In order to stop the running database instance run the following command. simply CTRL-C the running mongod process

9.7.6 Building a simple REST Service

In this section we will focus on creating a simple rest service. To organize our work we will create the following directory:

```
$ mkdir -p ~/cloudmesh/eve  
$ cd ~/cloudmesh/eve
```

As Eve needs a configuration and it is read in by default from the file settings.py we place the following content in the file ~/cloudmesh/eve/settings.py:

```
MONGO_HOST = 'localhost'  
MONGO_PORT = 27017  
MONGO_DBNAME = 'student_db'  
DOMAIN = {  
    'student': {  
        'schema': {  
            'firstname': {  
                'type': 'string'  
            },  
            'lastname': {  
                'type': 'string'  
            },  
            'university': {  
                'type': 'string'  
            },  
            'email': {  
                'type': 'string',  
                'unique': True  
            }  
            'username': {  
                'type': 'string',  
                'unique': True  
            }  
        }  
    }  
}  
RESOURCE_METHODS = ['GET', 'POST']
```

The DOMAIN object specifies the format of a `student` object that we are using as part of our REST service. In addition we can specify RESOURCE_METHODS which methods are activated for the REST service. This way the developer can restrict the available methods for a REST

service. To pass along the specification for mongoDB, we simply specify the hostname, the port, as well as the database name.

Now that we have defined the settings for our example service, we need to start it with a simple python program. We could name that program anything we like, but often it is called simply `run.py`. This file is placed in the same directory where you placed the **settings.py**. In our case it is in the file `~/cloudmesh/eve/run.py` and contains the following python program:

```
from eve import Eve
app = Eve()

if __name__ == '__main__':
    app.run()
```

This is the most minimal application for Eve, that uses the `settings.py` file for its configuration. Naturally, if we were to change the configuration file and for example change the DOMAIN and its schema, we would naturally have to remove the database previously created and start the service new. This is especially important as during the development phase we may frequently change the schema and the database. Thus it is convenient to develop necessary cleaning actions as part of a Makefile which we leave as easy exercise for the students.

Next, we need to start the services which can easily be achieved in a terminal while running the commands:

Previously we started the mongoDB service as follows:

```
$ mongod --dbpath ~/cloudmesh/data/db/
```

This is done in its own terminal, so we can observe the log messages easily. Next we start in another window the Eve service with

```
$ cd ~/cloudmesh/eve
$ python run.py
```

You can find the codes and commands up to this point in the following document.

9.7.7 Interacting with the REST service

Yet in another window, we can now interact with the REST service. We can use the commandline to save the data in the database using the REST api. The data can be retrieved in XML or in json format. Json is often more convenient for debugging as it is easier to read than XML.

Naturally, we need first to put some data into the server. Let us assume we add the user Albert Zweistein.

```
$ curl -H "Content-Type: application/json" -X POST \
-d '{"firstname":"Albert","lastname":"Zweistein", \
"school":"ISE","university":"Indiana University", \
"email":"albert@iu.edu", "username": "albert"}' \
http://127.0.0.1:5000/student/
```

To achieve this, we need to specify the header using **H** tag saying we need the data to be saved using json format. And **X** tag says the HTTP protocol and here we use POST method. And the tag **d** specifies the data and make sure you use json format to enter the data. Finally, the REST api endpoint to which we must save data. This allows us to save the data in a table called **student** in MongoDB within a database called **eve**.

In order to check if the entry was accepted in mongo and included in the server issue the following command sequence in another terminal:

```
$ mongo
```

Now you can query mongo directly with its shell interface

```
> show databases
> use student_db
> show tables # query the table names
> db.student.find().pretty() # pretty will show the json in a clear way
```

Naturally this is not really necessary for A REST service such as eve as we show you next how to gain access to the data via mongo while using REST calls. We can simply retrieve the information with the help of a simple URI:

```
$ curl http://127.0.0.1:5000/student?firstname=Albert
```

Naturally, you can formulate other URLs and query attributes that are passed along after the ?.

This will now allow you to develop sophisticated REST services. We encourage you to inspect the documentation provided by Eve to showcase additional features that you could be using as part of your efforts.

Let us explore how to properly use additional REST API calls. We assume you have MongoDB up and running. To query the service itself we can use the URI on the Eve port

```
$ curl -i http://127.0.0.1:5000
```

Your payload should look like the one below, if your output is not formatted like below try adding ?pretty=1

```
$ curl -i http://127.0.0.1:5000?pretty=1

HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 150
Server: Eve/0.7.6 Werkzeug/0.11.15 Python/2.7.5
Date: Wed, 17 Jan 2018 18:34:07 GMT
```

```
{
  "_links": {
    "child": [
      {
        "href": "student",
        "title": "student"
      }
    ]
  }
}
```

Remember that the API entry points include additional information such as links and a child, and href.

Set up a python environment that works for your platform. Provide explicit reasons why anaconda and other prepackaged python versions have issues for cloud related activities. When may you use anaconda and when should you not use anaconda. Why would you want to use pyenv?

What is the meaning and purpose of links, child, and href

In this case how many child resources are available through our API?

Develop a REST service with Eve and start and stop it

Define curl calls to store data into the service and retrieve it.

Write a Makefile and in it a target clean that cleans the data base. Develop additional targets such as start and stop, that start and stop the mongoDB but also the Eve REST service

Issue the command

```
$ curl -i http://127.0.0.1:5000/people
```

What does the `_links` section describe?

What does the `_items` section describe?

```
{
    "_items": [],
    "_links": {
        "self": {
            "href": "people",
            "title": "people"
        },
        "parent": {
            "href": "/",
            "title": "home"
        }
    },
    "_meta": {
        "max_results": 25,
        "total": 0,
        "page": 1
    }
}
```

9.7.8 Creating REST API Endpoints

Next we want to enhance our example a bit. First, let us get back to the eve working directory with

```
$ cd ~/cloudmesh/eve
```

Add the following content to a file called **run2.py**

```
from eve import Eve
from flask import jsonify
import os
import getpass
app = Eve()
@app.route('/student/albert')
def alberts_information():
    data = {
        'firstname': 'Albert',
        'lastname': 'Zweistsein',
        'university': 'Indiana University',
        'email': 'albert@example.com'
    }
    try:
        data['username'] = getpass.getuser()
    except:
        data['username'] = 'not-found'
    return jsonify(**data)

if __name__ == '__main__':
    app.run(debug=True, host="127.0.0.1")
```

After creating and saving the file. Run the following command to start the service

```
$ python run2.py
```

After running the command, you can interact with the service while entering the following url in the web browser:

```
http://127.0.0.1:5000/student/alberts
```

You can also open up a second terminal and type in it

```
$ curl http://127.0.0.1:5000/student/alberts
```

The following information will be returned:

```
{
    "firstname": "Albert",
    "lastname": "Zweistain",
    "university": "Indiana University",
    "email": "albert@example.com",
    "username": "albert"
}
```

This example illustrates how easy it is to create REST services in python while combining information from a dict with information

retrieved from the system. The important part is to understand the decorator **app.route**. The parameter specifies the route of the API endpoint which will be the address appended to the base path, `http://127.0.0.1:5000`. It is important that we return a jsonified object, which can easily be done with the `jsonify` function provided by flask. As you can see the name of the decorated function can be anything you like. The route specifies how we access it from the service.

9.7.9 REST API Output Formats and Request Processing

Another way of managing the data is to utilize class definitions and response types that we explicitly define.

If we want to create an object like Student, we can first define a python class. Create a file called **student.py**. Please, note the get method that returns simply the information in the dict for the class. It is not related to the REST get function.

```
class Student(object):
    def __init__(self, firstname, lastname, university, email):
        self.firstname = firstname
        self.lastname = lastname
        self.university = university
        self.email = email
        self.username = 'undefined'
    def get(self):
        return self.__dict__
    def setUsername(self, name):
        self.username = name
        return name
```

Next we define a REST service with Eve as shown in the following listing

```
from eve import Eve
from student import Student
import platform
import psutil
import json
from flask import Response
import getpass
app = Eve()
@app.route('/student/albert', methods=['GET'])
def processor():
    student = Student("Albert",
                      "Zweistein",
                      "Indiana University",
```

```

        "albert@example.edu")

response = Response()
response.headers["Content-Type"] = "application/json; charset=utf-8"

try:
    student.setUsername(getpass.getuser())
    response.headers["status"] = 200
except:
    response.headers["status"] = 500

response.data = json.dumps(student.get())
return response

if __name__ == '__main__':
    app.run(debug=True, host='127.0.0.1')

```

In contrast to our earlier example, we are not using the jsonify object, but create explicitly a response that we return to the clients. The response includes a header that we return the information in json format, a status of 200, which means the object was returned successfully, and the actual data.

9.7.10 REST API Using a Client Application

○ This example is not tested. Please provide feedback and improve.

In the Section [Rest Services with Eve](#) we created our own REST API application using Python Eve. Now once the service running, a we need to learn how to interact with it through clients.

First go back to the working folder:

```
$ cd ~/cloudmesh/eve
```

Here we create a new python file called **client.py**. The file include the following content.

```

import requests
import json

def get_all():
    response = requests.get("http://127.0.0.1:5000/student")
    print(json.dumps(response.json(), indent=4, sort_keys=True))

def save_record():

```

```

headers = {
    'Content-Type': 'application/json'
}

data = '{"firstname": "Gregor",
         "lastname": "von Laszewski",
         "university": "Indiana University",
         "email": "jane@iu.edu",
         "username": "jane"}'

response = requests.post('http://localhost:5000/student/',
                         headers=headers,
                         data=data)
print(response.json())

if __name__ == '__main__':
    save_record()
    get_all()

```

Run the following command in a new terminal to execute the simple client by

```
$ python client.py
```

Here when you run this class for the first time, it will run successfully, but if you tried it for the second time, it will give you an error. Because we did set the email to be a unique field in the schema when we designed the settings.py file in the beginning. So if you want to save another record you must have entries with unique emails. In order to make this dynamic you can include a input reading by using the terminal to get the student data first and instead of the static data you can use the user input data from the terminal to get dynamic data. But for this exercise we do not expect that or any other form data functionality.

In order to get the saved data, you can comment the record saving function and uncomment the get all function. In python commenting is done by using #.

This client is using the **requests** python library to send GET, POST and other HTTP requests to the server so you can leverage build in methods to simplify your work.

The `get_all` function provides a way to get the output to the console

with all the data in the student database. The `save_record` function provides a way to save data in the database. You can create dynamic functions in order to save dynamic data. However it may take some time for you to apply as exercise.

Write a RESTful service to determine a useful piece of information off of your computer i.e. disk space, memory, RAM, etc. In this exercise what you need to do is use a python library to extract data about computer information mentioned previously and send these information to the user once the user calls an API endpoint like `http://localhost:5000/performance/ram`, it must return the RAM value of the given machine. For each information like disk space, RAM, etc you can use an endpoint per each feature needed. As a tip for this exercise, use the psutil library in python to retrieve the data, and then get these information into an string then populate a class called Computer and try to save the object like wise.

9.7.11 Towards cmd5 extensions to manage eve and mongo

⚠ part of this section related to management of the mongo db serviceis done by the cm4 command we will be developping as part of this class `cms mongo admin` that does all of the things explained next and more.

Naturally it is of advantage to have in cms administration commands to manage mongo and eve from cmd instead of targets in the Makefile. Hence, we **propose** that the class develops such an extension. We will create in the repository the extension called admin and hope that students through collaborative work and pull requests complete such an admin command.

The proposed command is located at:

- <https://github.com/cloudmesh/cloudmesh.rest/blob/master/cl>

It will be up to the class to implement such a command. Please coordinate with each other.

The implementation based on what we provided in the Make file seems straight forward. A great extension is to load the objects definitions or eve e.g. settings.py not from the class, but from a place in .cloudmesh. I propose to place the file at:

```
~/cloudmesh/db/settings.py
```

the location of this file is used when the Service class is initialized with None. Prior to starting the service the file needs to be copied there. This could be achieved with a set command.

9.8 HATEOAS



In the previous section we discussed the basic concepts about RESTful web service. Next we introduce you to the concept of HATEOAS

HATEOAS stands for Hypermedia as the Engine of Application State and this is enabled by the default configuration within Eve. It is useful to review the terminology and attributes used as part of this configuration. HATEOAS explains how REST API endpoints are defined and it provides a clear description on how the API can be consumed through these terms:

_links

Links describe the relation of current resource being accessed to the rest of the resources. It is like if we have a set of links to the set of objects or service endpoints that we are referring in the RESTful web service. Here an endpoint refers to a service call which is responsible for executing one of the CRUD operations on a particular object or set of objects. More on the links, the links object contains the list of serviceable API endpoints or list of services. When we are calling a GET request or any other request, we can use these service endpoints to execute different queries based on the user purpose. For instance, a service call can be used to insert data or retrieve data from a remote database using a REST API call. About databases we will discuss in detail in

another chapter.

title

The title in the rest endpoint is the name or topic that we are trying to address. It describes the nature of the object by a single word. For instance student, bank-statement, salary,etc can be a title.

parent

The term parent refers to the very initial link or an API endpoint in a particular RESTful web service. Generally this is denoted with the primary address like <http://example.com/api/v1/>.

href

The term href refers to the url segment that we use to access the a particular REST API endpoint. For instance “student?page=1” will return the first page of student list by retrieving a particular number of items from a remote database or a remote data source. The full url will look like this, “<http://www.exampleapi.com/student?page=1>”.

In addition to these fields eve will automatically create the following information when resources are created as showcased ot

- <http://python-eve.org/features.html>

Field	Description
<code>created</code>	item creation date.
<code>updated</code>	item last updated on.
<code>etag</code>	ETag, to be used for concurrency control and conditional requests.
<code>_id</code>	unique item key, also needed to access the individual item endpoint.

Pagenation information can be included in the `_meta` field.

9.8.1 Filtering

Clients can submit query strings to the rest service to retrieve resources based on a filter. This also allows sorting of the results queried. One nice feature about using mongo as a backend database is that Eve not only allows python conditional expressions, but also mongo queries.

A number of examples to conduct such queries include:

```
$ curl -i -g http://eve-demo.herokuapp.com/people?where={%22lastname%22:%20%22Doe%22}
```

A python expression

```
$ curl -i http://eve-demo.herokuapp.com/people?where=lastname=="Doe"
```

9.8.2 Pretty Printing

Pretty printing is typically supported by adding the parameter `?pretty` or `?pretty=1`

If this does not work you can always use python to beautify a json output with

```
$ curl -i http://localhost/people?pretty
```

or

```
$ curl -i http://localhost/people | python -m json.tool
```

9.8.3 XML

If for some reason you like to retrieve the information in XML you can specify this for example through curl with an Accept header

```
$ curl -H "Accept: application/xml" -i http://localhost
```

9.9 EXTENSIONS TO EVE



A number of extensions have been developed by the community. This includes eve-swagger, eve-sqlalchemy, eve-elastic, eve-mongoengine, eve-neo4j, eve.net, eve-auth-jwt, and flask-sentinel.

Naturally there are many more.

Students have the opportunity to pick one of the community extensions and provide a section for the handbook.

Pick one of the extension, research it and provide a small section for the handbook so we add it.

9.9.1 Object Management with Eve and Evegenie

<http://python-eve.org/>

Eve makes the creation of a REST implementation in python easy. We will provide you with an implementation example that showcases that we can create REST services without writing a single line of code. The code for this is located at <https://github.com/cloudmesh/rest>

This code will have a master branch but will also have a dev branch in which we will add gradually more objects. Objects in the dev branch will include:

- virtual directories
- virtual clusters
- job sequences
- inventories

You may want to check our active development work in the dev branch. However for the purpose of this class the master branch will be sufficient.

9.9.1.1 Installation

First we have to install mongodb. The installation will depend on your operating system. For the use of the rest service it is not important to

integrate mongodb into the system upon reboot, which is focus of many online documents. However, for us it is better if we can start and stop the services explicitly for now.

On ubuntu, you need to do the following steps:

○ TODO: Section can be contributed by student.

On windows 10, you need to do the following steps:

○ TODO: Section can be contributed by student. If you elect Windows 10. You could be using the online documentation provided by starting it on Windows, or running it in a docker container.

On macOS you can use home-brew and install it with:

```
$ brew update  
$ brew install mongodb
```

In future we may want to add ssl authentication in which case you may need to install it as follows:

```
$ brew install mongodb --with-openssl
```

9.9.1.2 Starting the service

We have provided a convenient Makefile that currently only works for macOS. It will be easy for you to adapt it to Linux. Certainly you can look at the targets in the makefile and replicate them one by one. Important targets are deploy and test.

When using the makefile you can start the services with:

```
$ make deploy
```

IT will start two terminals. IN one you will see the mongo service, in the other you will see the eve service. The eve service will take a file called sample.settings.py that is base on sample.json for the start of the eve service. The mongo service is configured in such a way that it only accepts incoming connections from the local host which will be

sufficient for our case. The mongo data is written into the `$USER/.cloudmesh` directory, so make sure it exists.

To test the services you can say:

```
$ make test
```

You will see a number of json text been written to the screen.

9.9.1.3 Creating your own objects

The example demonstrated how easy it is to create a mongodb and an eve rest service. Now let us use this example to create your own. For this we have modified a tool called eogenie to install it onto your system.

The original documentation for eogenie is located at:

- <http://eogenie.readthedocs.io/en/latest/>

However, we have improved eogenie while providing a commandline tool based on it. The improved code is located at:

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

You clone it and install on your system as follows:

```
$ cd ~/github
$ git clone https://github.com/cloudmesh/eogenie
$ cd eogenie
$ python setup.py install
$ pip install .
```

This should install in your system eogenie. You can verify this by typing:

```
$ which eogenie
```

If you see the path eogenie is installed. With eogenie installed its usage is simple:

```
$ eogenie
```

```
Usage:  
evegenie --help  
evegenie FILENAME
```

It takes a json file as input and writes out a settings file for the use in eve. Lets assume the file is called sample.json, than the settings file will be called sample.settings.py. Having the evegenie program will allow us to generate the settings files easily. You can include them into your project and leverage the Makefile targets to start the services in your project. In case you generate new objects, make sure you rerun evegenie, kill all previous windows in which you run eve and mongo and restart. In case of changes to objects that you have designed and run previously, you need to also delete the mongod database.

9.10 DJANGO REST FRAMEWORK



Django REST framework is a large toolkit to develop Web APIs. The developers of the framework provide the following reasons for using it:

“(1) The Web browsable API is a huge usability win for your developers. (2) Authentication policies including packages for OAuth1a and OAuth2. (3) Serialization that supports both ORM and non-ORM data sources. (4) Customizable all the way down - just use regular function-based views if you do not need the more powerful features. (5) Extensive documentation, and great community support. (6) Used and trusted by internationally recognised companies including Mozilla, Red Hat, Heroku, and Eventbrite.”

9.11 GITHUB REST SERVICES



In this section we want to explore a more features of REST services and how to access them. Naturally many cloud services provide such REST sinterfaces. This is valid for IaaS, PaaS, and SaaS.

Instead of using a REST service for IaaS, let us here inspect a REST service for the Github.com platform.

Its interfaces are documented nicely at

- <https://developer.github.com/v3/>

We see that Github offers many resources that can be accessed by te users which includes

- Activities
- Checks
- Gists
- Git Data
- GitHub Apps
- Issues
- Migrations
- Miscellaneous
- Organizations
- Projects
- Pull Requests
- Reactions
- Repositories
- Searches
- Teams
- Users

Most likely we forgot the one or the other Resource that we can access via REST. It will be out of scope for us to explore all of these issues, so let us focus on how we for example access Github Issues. In fact we will use the script that we use to create issue tables for this book to showcase how easy the interaction is and to retrieve the information.

9.11.1 Issues

The REST service for issues is described in the following Web page as specification

- <https://developer.github.com/v3/issues/>

We see the following functionality:

- [List issues](#)
- [List issues for a repository](#)
- [Get a single issue](#)
- [Create an issue](#)
- [Edit an issue](#)
- [Lock an issue](#)
- [Unlock an issue](#)
- [Custom media types](#)

As we have learned in our REST section we need to issue GET requests to obtain information about the issues. Such as

```
GET /issues
GET /user/issues
```

As response we obtain a json object with the information we need to further process it. Unfortunately, the free tier of github has limitations in regards to the frequency we can issue such requests to the service, as well as in the volume in regards to number of pages returned to us.

Let us now explore how to easily query some information. In our example we like to retrieve the list of issues for a repository as LaTeX table but also as markdown. This way we can conveniently integrate it in documents of either format. As LaTeX has a more sophisticated table management, let us first create a LaTeX table document and then use a program to convert LaTeX to markdown. For the latter we can reuse a program called `pandoc` that can convert the table from LaTeX to markdown.

Let us assume we have a program called `issues.py` that prints the table in markdown format

```
$ python issues.py
```

An example for such a program is listed at.

- <https://github.com/cloudmesh-community/book/blob/master/bin/issues.py>

Although python provides the very nice module `requests` which we typically use for such issues. we have here just wrapped the commandline call to `curl` into a system command and redirect its output to a file. However, as we only get limited information back in pages, we need to continue such a request multiple times. to keep things simple we identified that for the project at this time not more than n pages need to be fetched, so we append the output from each page to the file.

Your task is it to improve this script and automatize this activity so that no maximum fetches have to be entered.

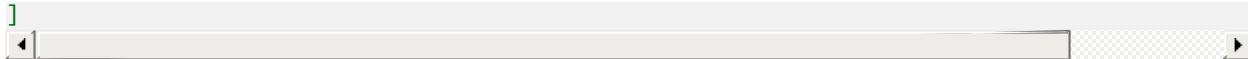
The reason why this program is so short is that we leverage the build in function for `json` data structure manipulation, hear a read and a dump. When we look in the `issue.json` file that is created as intermediary file we see a list of items such as

```
[  
...  
{  
    "url": "https://api.github.com/repos/cloudmesh-community/book/issues/46",  
    "repository_url": "https://api.github.com/repos/cloudmesh-community/book",  
    "labels_url": "https://api.github.com/repos/cloudmesh-community/book/issues/46/labels",  
    "comments_url": "https://api.github.com/repos/cloudmesh-community/book/issues/46/comments",  
    "events_url": "https://api.github.com/repos/cloudmesh-community/book/issues/46/events",  
    "html_url": "https://github.com/cloudmesh-community/book/issues/46",  
    "id": 360613438,  
    "node_id": "MDU6SXNzdWUzNjA2MTM0Mzg=",  
    "number": 46,  
    "title": "Taken: Virtualization",  
    "user": {  
        "login": "laszewsk",  
        "id": 425045,  
        "node_id": "MDQ6VXNlcjQyNTA0NQ==",  
        "avatar_url": "https://avatars1.githubusercontent.com/u/425045?v=4",  
        "gravatar_id": "",  
        "url": "https://api.github.com/users/laszewsk",  
        "html_url": "https://github.com/laszewsk",  
        "followers_url": "https://api.github.com/users/laszewsk/followers",  
        "following_url": "https://api.github.com/users/laszewsk/following{/other_user}",  
        "gists_url": "https://api.github.com/users/laszewsk/gists{/gist_id}",  
        "starred_url": "https://api.github.com/users/laszewsk/starred{/owner}{/repo}",  
        "subscriptions_url": "https://api.github.com/users/laszewsk/subscriptions",  
        "organizations_url": "https://api.github.com/users/laszewsk/orgs",  
        "repos_url": "https://api.github.com/users/laszewsk/repos",  
    }  
}
```

```

        "events_url": "https://api.github.com/users/laszewsk/events{/privacy}",
        "received_events_url": "https://api.github.com/users/laszewsk/received_events",
        "type": "User",
        "site_admin": false
    },
    "labels": [],
    "state": "open",
    "locked": false,
    "assignee": {
        "login": "laszewsk",
        "id": 425045,
        "node_id": "MDQ6VXNlcjQyNTA0NQ==",
        "avatar_url": "https://avatars1.githubusercontent.com/u/425045?v=4",
        "gravatar_id": "",
        "url": "https://api.github.com/users/laszewsk",
        "html_url": "https://github.com/laszewsk",
        "followers_url": "https://api.github.com/users/laszewsk/followers",
        "following_url": "https://api.github.com/users/laszewsk/following{/other_user}",
        "gists_url": "https://api.github.com/users/laszewsk/gists{/gist_id}",
        "starred_url": "https://api.github.com/users/laszewsk/starred{/owner}{/repo}",
        "subscriptions_url": "https://api.github.com/users/laszewsk/subscriptions",
        "organizations_url": "https://api.github.com/users/laszewsk/orgs",
        "repos_url": "https://api.github.com/users/laszewsk/repos",
        "events_url": "https://api.github.com/users/laszewsk/events{/privacy}",
        "received_events_url": "https://api.github.com/users/laszewsk/received_events",
        "type": "User",
        "site_admin": false
    },
    "assignees": [
        {
            "login": "laszewsk",
            "id": 425045,
            "node_id": "MDQ6VXNlcjQyNTA0NQ==",
            "avatar_url": "https://avatars1.githubusercontent.com/u/425045?v=4",
            "gravatar_id": "",
            "url": "https://api.github.com/users/laszewsk",
            "html_url": "https://github.com/laszewsk",
            "followers_url": "https://api.github.com/users/laszewsk/followers",
            "following_url": "https://api.github.com/users/laszewsk/following{/other_user}",
            "gists_url": "https://api.github.com/users/laszewsk/gists{/gist_id}",
            "starred_url": "https://api.github.com/users/laszewsk/starred{/owner}{/repo}",
            "subscriptions_url": "https://api.github.com/users/laszewsk/subscriptions",
            "organizations_url": "https://api.github.com/users/laszewsk/orgs",
            "repos_url": "https://api.github.com/users/laszewsk/repos",
            "events_url": "https://api.github.com/users/laszewsk/events{/privacy}",
            "received_events_url": "https://api.github.com/users/laszewsk/received_events",
            "type": "User",
            "site_admin": false
        }
    ],
    "milestone": null,
    "comments": 0,
    "created_at": "2018-09-16T07:35:35Z",
    "updated_at": "2018-09-16T07:35:35Z",
    "closed_at": null,
    "author_association": "CONTRIBUTOR",
    "body": "Develop a section about Virtualization"
},
...

```



As we can see from this entry there is a lot of information associated that for our purposes we do not need, but certainly could be used to mine github in general.

We like to point out that github is actively mined for exploits where passwords are posted in clear text for AWS, Azure and other clouds. This is a common mistake as many sample programs ask the student to place the password directly into their programs instead of using a configuration file that is never part of the code repository.

9.11.2 Exercise

E.github.issues.1:

Develop a new code like the one in this section, but use python requests instead of the `os.system` call.

E.github.issues.2:

In the simple program we hardcoded the number of page requests. How can we find out exactly how many pages we need to retrieve? Implement your solution

E.github.issues.3:

Be inspired by the many REST interfaces. Who can they be used to mine interesting things.

E.github.issues.4:

Can you create a project, author, or technology map based on information that is available in github. For example python projects may include a requirements file, or developers may work on some projects together, but others do other projects with others can you create a network?

E.github.issues.5:

Use github to develop some cool python programs that show some statistics about github. An example would be: Given a github repository, show the checkins by date and visualize them graphically for one committer and all committers. Use bokeh or matplotlib.

E.github.issues.6:

Develop a python program that retrieves a file. Develop a python program that uploads a file. Develop a class that does this and use it in your program. Use docopt to create a manual page. Please remember this prepares you for your project so this is very useful to do.



◆ Learning Objectives

- Gain understanding of the basic the concepts of virtualization
- Understand what a virtual machine is
- Understand what a Hypervisor is

Virtualization is one of the important technologies that started the cloud revolution. It provides the basic underlying principles for the development and adoption of clouds. The concept, although old and already used in the early days of computing, has recently been exploited to lead to better utilization of servers as part of data centers, but also the local desktops.

Virtualization enables to execute multiple applications in such a way that the applications seem to run independently form each other in their own virtualized context.

Examples of the usefulness of virtualization include testing of applications and run experiments on a different operating system than the one on our host computer. To enable this we need virtual machines.

10.1 VIRTUAL MACHINES

We define a virtual machine as follows:

A virtual machine (VM) is a software-based emulation of a computer system. This can include process virtualization and physical computer virtualization such as running an operating system. Multiple virtual machines share the resources of the computer or system on which they run.

We distinguish the following types of Virtual machines

- **System Virtual Machines** or **Hardware Virtual Machine**, which is the virtualization of the operating system providing a complete system platform environment emulating the hardware. Here we essentially run another operating system on top of the existing OS while using a software abstraction between them allowing the virtualization.

Examples of such virtualization include VirtualBox and VMWare.

- **Process Virtual Machines** or **Application Virtual Machine** which provides a platform independent programming environment that abstracts the details of the underneath hardware or OS from software or application runtime.

Examples of such virtual machines include Java Virtual Machine (JVM) and the .NET Framework

Next we will be analyzing the system machine virtualization in more detail, as they are one of the reasons for the clouds revolution.

10.2 SYSTEM VIRTUAL MACHINES

The use of a system as a virtual machine has its clear advantages for the cloud. We distinguish two main ways of system virtualizing:

- **Bare-metal Virtualization** in which the virtual machine monitor is installed directly on top of the hardware so that it has direct access to the underlying hardware. It hosts the operating system. The VMM is also called hypervisor. We also use for bare-metal supporting VMM the term Type 1 hypervisor.
- **Hosted Virtualization** in which the base operating system is installed on the hardware. Here a virtual machine monitor (VMM) is installed on top of the host OS allowing the users to

run other operating systems on the VMM. In addition, the Virtual Machine Monitor or Hypervisor manages the deployments of potentially multiple virtual machines on top of the underlying Operating system. We also use for hosted VMM the term Type 2 hypervisor.

In either case the functionality a virtual machine is supported through configuration files, specifications, and access to the physical resources either directly or indirectly through the host OS. A virtual machine provides the same functionality as a physical computer, but with the advantage that through virtualization they are portable, can be managed and provide increased security while shielding the underlying OS from harmful actions. As a virtual machine is in principle a program, it consists of several files including a configuration file, virtual disk files, virtual RAM, and a log file. Virtual machines are configured to run a virtual operating system that allows applications to run on them. Each virtual machine has its own copy of the OS making it independent and more secure.

End users and developers will benefit from using virtual machines in case they need to operate or support on different hardware or porting software on it.

10.3 HOSTED VIRTUALIZATION

As in the hosted virtualization the guest operating system accessed the underlying hardware through the host OS, it usually has limited access to the hardware as defined by the host OS. This allows the host OS to impose policies that govern the operation of multiple guest OS concurrently. This includes management and scheduling of processes, memory, I/O operations to assign them appropriately to the guest OS. Through this mechanism the hypervisor provides an emulation of available hardware to each Virtual Machine run on top of it in time-sharing fashion for resource constrained or resource shared activities.

As example, the hypervisor has the ability to present generic I/O

devices and it has no access to non-generic I/O devices. Generic I/O devices are network, interface cards, CD-ROMs. Examples for non-generic I/O devices are PCI data acquisition card, etc. However with appropriate driver support even such devices could be made accessible to the VMs.

Often we also find that hosted virtualization supports connected USB drives in the VMs which becomes very practical for USB attached devices needed in storage, or even edge computing applications.

Advantages of Hosted Virtualization include

- Multiple Operating systems run on separate virtual machines on a VMM.
- Different Operating systems run on separate virtual machines on a VMM.
- Hardware level driver support is controlled by VMM, allowing an isolation of certain security aspects for accessing the hardware.
- Installation of software can be done by the owner of the virtual machine and does not have to be conducted by the provider of the hypervisor.

Disadvantages of Hosted Virtualization include

- Increased resource requirements as the Guest OS is running a full copy of the OS. In its worst case this will lead to a significant performance reduction while using resources that are in contention.
- The user of hypervisors must be familiar with operating system management and security to ensure it is safe to use.

10.4 SUMMARY

To showcase how these technologies relate to each other you may review [Figure 30](#)

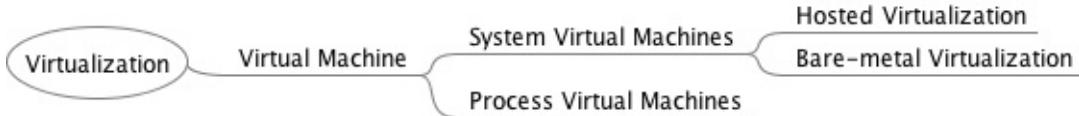


Figure 30: Virtualization Taxonomy

We summarize the following hypervisor types:

- Type-1 hypervisors supporting native or bare-metal. They run directly on the host's hardware to control the hardware and to manage guest operating systems.
- Type-2 hypervisors supporting hosted virtualization. They run on a conventional operating system (OS) just as other computer programs do. A guest operating system runs as a process on the host.

10.5 VIRTUALIZATION APPROACHES

Next we look at different virtualization approaches that relate to resource utilization.

10.5.1 Full virtualization

When looking at virtualization we often identify it with being a full virtualization. The hypervisor provides a full abstraction of the hardware exposed to the guest OSs. In this case, the guest OSs the virtual machine just run without any special modification on the host OS. It just looks like an independent running computer [22].

10.5.2 Paravirtualization

Para – alongside/partial – virtualization is developed to improve performance by interacting between the OS and the hypervisor. This is done for complex and time-consuming tasks that otherwise could not be managed by the VMM manager. Commands sent from the OS to the hypervisor are called hypercalls [22].

10.6 VIRTUALIZATION TECHNOLOGIES

In this section we cover introduction to underlying virtualization technologies used on some mainstream platforms.

Cloud providers, such as AWS, Azure, and Google, and OpenStack use for example QEMU and KVM technologies for compute instance virtualization.

10.6.1 Selected Hardware Virtualization Technologies

10.6.2 AMD-V and Intel-VT

The hardware virtualization support enabled by AMD-V and Intel VT technologies introduces virtualization in the x86 processor architecture. According to Intel, Intel Hyper-Threading Technology allows a single processor to execute two or more separate threads concurrently. When it is enabled, multi-threaded software applications can execute their threads in parallel, thereby improving their performance.

10.6.3 I/O MMU virtualization (AMD-Vi and Intel VT-d)

The term IOMMU is an abbreviation for input-output memory management unit. An IOMMU allows through virtual addresses to interface with physical addresses, allowing external direct-memory-access-capable IO devices to interface with the main memory [23]. AMD's I/O Virtualization Technology (AMD-Vi) was originally called IOMMU.

To use Intel's Virtualization Technology for Directed I/O (VT-d), both the motherboard chipset and system firmware (BIOS or UEFI) need to fully support the IOMMU I/O virtualization functionality for it to be usable [24].

10.6.4 Selected VM Virtualization Software and Tools

A number of noteworthy virtualization software and tools exist which make the development and use of virtualization on the hardware possible. They include

- Libvirt
- KVM
- Xen
- Hyper-V
- QEMU
- VMWare
- VirtualBox

We will be discussing them next.

10.6.4.1 Libvirt

[Libvirt](#) is an library with an API for managing virtualization solutions such as provided by KVM and Xen. It provides a common management API for them, allowing uniform, cross-hypervisor interfaces for higher-level management tools. `Libvirt` provides a toolkit to manage virtualization hosts and supports a wide set of languages, such as C, Python, Perl, and Java. Drivers are the basic building block for libvirt functionality to support the capability to handle specific hypervisor driver calls. Drivers are discovered and registered during connection processing as part of the `virInitializeAPI`. Each driver has a registration API which loads up the driver specific function references for the libvirt APIs to call. The following is a simplistic view of the hypervisor driver mechanism. Furthermore, it provides APIs for management of virtual networks and storage on the VM Host Server. The configuration of each VM Guest is stored in an XML file [25]. The official website for `libvirt` is located at

- <https://libvirt.org/>

10.6.4.2 QEMU

QEMU is a virtualization technology emulator that allows you to run

operating systems and Linux distributions on your current system without installing them or burn their ISO files. When used as a machine emulator, QEMU can run OSs and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance. QEMU provides two generic functions. One of them is open source machine emulator and the other is a virtualizer.

- Machine emulation: using it as a machine emulator it runs the OSs and programs designed for one machine on a different machine of potential different architecture. It uses dynamic translation through which it achieves very good performance.
- Virtualizer: Using it as a virtualizer it executes the guest code directly on the host CPU. This enables QEMU to achieve near native performance.

Once QEMU has been installed, it should be ready to run a guest OS from a disk image. This image is a file that represents the filesystem and OS on a hard disk. From the perspective of the guest OS, it actually is a file on hard disk, and it can create its own filesystem on the virtual disk.

QEMU supports either XEN or KVM to enable virtualization. With the help of KVM, QEMU can virtualize x86, server and embedded PowerPC, 64-bit POWER, S390, 32-bit and 64-bit ARM, and MIPS guests according to the [QEMU Wiki](#).

Useful links include the following:

- An extensive manual is provided at <https://qemu.weilnetz.de/doc/qemu-doc.html>.
- QEMU can be downloaded from <http://www.qemu.org/download/>.
- A collection of images for testing purposes is provided at https://wiki.qemu.org/Testing/System_Images

An example for using QEMU is provided in Section [Virtual Machine Management with QEMU]{???

10.6.4.3 KVM

KVM, or Kernel-based Virtual Machine is a popular open-source hypervisor solution. It was released as a virtualization solution for Linux based systems, and later was merged into Linux Kernel since version 2.6.20. It was originally supporting x86 hardware with virtualization extensions (Intel VT or AMD-V), but later supporting of PowerPC and ARM were added. It supports a variety of different guest OSs, e.g., Windows family, Darwin (the core of MacOS), in addition to the different distros from various linux operating systems. The full supported guest list can be found at: http://www.linux-kvm.org/page/Guest_Support_Status

The full list of KVM features can be found here: http://www.linux-kvm.org/page/KVM_Features. Among them, some cool features include hot-plugging of hardware even CPU and PCI devices. It supports live migration of VMs too.

10.6.4.3.1 KVM vs QEMU

KVM includes a fork of the QEMU executable. The QEMU project focuses on hardware emulation and portability. KVM focus on the kernel module and interfacing with the rest of the userspace code. KVM comes with a `kvm-qemu` executable that just like QEMU manages the resources while allocating RAM, loading the code. However instead of recompiling the code it spawns a thread which calls the KVM kernel module to switch to guest mode. It then proceeds to execute the VM code. When privileged instructions are found, it switches back to the KVM kernel module, and if necessary, signals the QEMU thread to handle most of the hardware emulation. This means that the guest code is emulated in a posix thread which can be managed with common Linux tools [26].

10.6.4.4 Xen

Xen is one of the most widely adopted hypervisors by IaaS cloud. It is supported by the earliest and still the most popular public cloud offering, i.e., Amazon Web Service (AWS). Eucalyptus, one open-source effort to replicate what AWS had to offer, and the then most popular private cloud software, supported Xen from the start. And later Openstack, the most popular open-source IaaS cloud software at present, also supports Xen.

Some notable features of Xen includes:

- Supporting x86-64 and ARM for host architecture.
- Supporting live migration of VMs between different physical hosts without losing the availability.

More detailed list can be found at https://wiki.xenproject.org/wiki/Xen_Project_Release_Features.

10.6.4.5 Hyper-V

Hyper-V is a product from Microsoft to support virtualization on systems running Windows. Hyper-V was originally released along with Windows Server 2008, with a separate free version with limited functionality. In later releases it adds more features, e.g., better support of Linux guest OS, live migration of VMs, etc.

Hyper-V is still getting a lot of popularity comparing to XEN and KVM which we attribute to the increasing presence of Microsoft's Azure cloud offering.

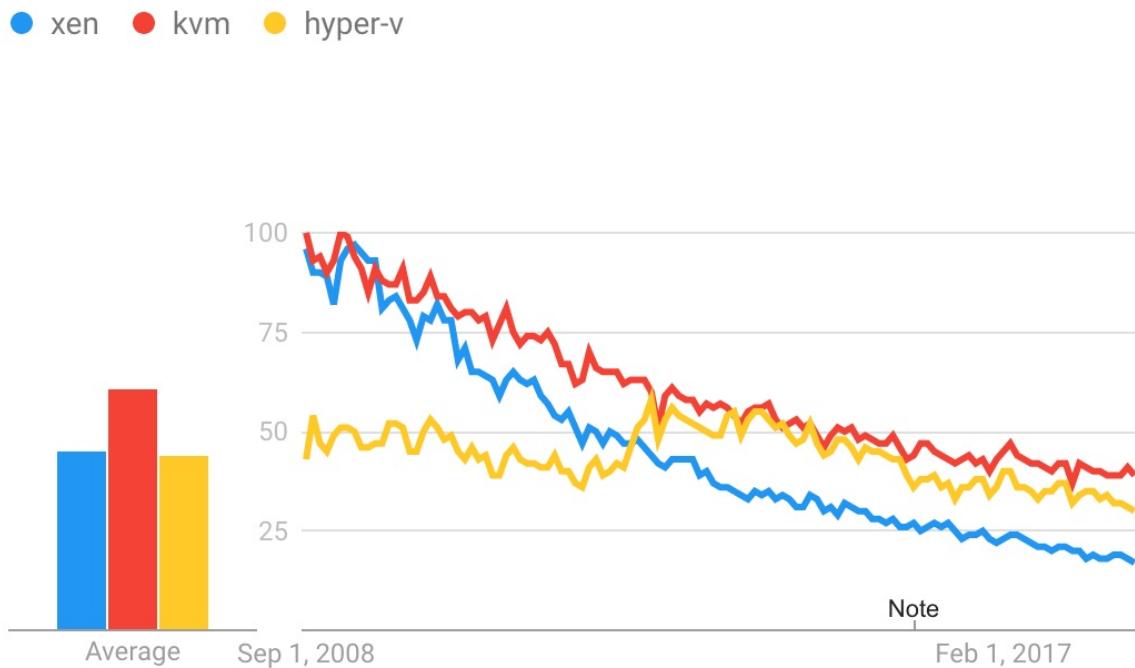


Figure 31: Popularity of KVM, Xen, and Hyper-V according to Google Trends

However overall the search popularity of hypervisors have been decreasing, as other lightweight virtualization solutions, i.e., container technologies become more main stream. We will covered them in a later chapter.

More detailed information about Hyper-V can be found at <https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-architecture>

10.6.4.6 VMWare

VMware is well known for the company bringing hypervisors to the amss market. The company is now owned by Dell. It has developed the first type 2 hypervisor. Today VMWare offer type 1 hypervisors and type 2 hypervisors [27].

Because the initial software virtualized “hardware for a video adapter, a network adapter, and hard disk adapters” as well as “pass-through drivers for guest USB, serial, and parallel devices” [27] it provided an

attractive solution for many to use it to run different OSs on their host computers. One important advantage is that it does not rely on virtualization extensions to the x86 instruction set as it was developed before they became available. This means it can run on many other platforms. However this advantage is diminished with the ubiquitous availability of these features in the hardware.

10.6.5 Parallels

Another interesting company offering hypervisors is Parallels. This company has two main products in that regards:

- Parallels Desktop for Mac, which for x86 machines allows users to run virtual machines independently using Windows, Linux, Solaris.
- Parallels Workstation for Microsoft Windows and Linux users which for x86 machines allows user to run virtual machines independently on the Windows host.

10.6.5.1 VirtualBox

VirtualBox is a free open-source hypervisor for x86 architectures. It is now owned by Oracle while transitioning from SUN which in turn acquired the original technology from Innotek.

One of the nice features for us is that VirtualBox is able to create and manage guest virtual machines such as Windows, Linux, BSD, OSx86 and even in part also macOS (on Apple hardware). Hence it makes it for us a very valuable tool while being able to run virtual machines on a local desktop or computer to simulate cloud resources without charging cost. In addition we find command line tools such as vagrant (see Section ?? O) that make the use convenient while not having to utilize the GUI or the more complex virtual box command interfaces. A guest additions package allows compatibility with the host OS, to for example allow window management between host and guest OS.

In Section [VirtualBox](#) we have provided a practical introduction to VirtualBox.

10.6.5.2 Wine - Wine is not an emulator

The next software that we introduce is actually not a hypervisor. However it is very interesting as a contrast to the other approach. The term Wine has been originally introduced as an acronym for Wine Is Not an Emulator. In contrast to the other approaches Wine introduces a compatibility layer that allows to run Windows applications on a number of POSIX-compliant operating systems. This includes Linux, macOS and BSD. In contrast to using a virtual machine or emulator, Wine translates Windows API calls into POSIX calls [28]. Hence, it allows to pass the Windows API calls directly to operating system calls leading to good performance [28]. The disadvantage of this approach is that in the early days and till today some of the underlying calls may not be ported yet and may lead to applications not running. Those of us that wanted to run for example Microsoft Word on Linux or macOS, may remember that Wine was the tool we used to do so even before Word was released on macOS.

10.6.5.3 Comparison of some technologies

QEMU and KVM are better integrated in Linux and has a smaller footprint. This may result in better performance. VirtualBox is targeted as virtualization software and limited to x86 and amd64. As Xen uses QEMU it allows hardware virtualization. However, Xen can also use paravirtualization [29]. In the following table we summarize support for full- and paravirtualization

	XEN	KVM	VirtualBox	VMWare
Paravirtualization	yes	no	no	no
Full virtualization	yes	yes	yes	yes

10.6.6 Selected Storage Virtualization Software and Tools

Storage virtualization allows the system to integrate the logical view of the physical storage resources into a single pool of storage. Users are unaware while using virtual storage that it is not hosted on the same hardware resources, such as disks. Storage virtualization is done across the various layers that build state of the art storage infrastructures. This includes Storage devices, the Block aggregation layer, the File/record layer, and the Application layer. Most recently hosting files as part of the application layer in the cloud is changing how we approach data storage needs in the enterprise. A good example for a cloud based virtual storage is google drive. Other systems include Box, AWS3 and Azure.

10.6.7 Selected Network Virtualization Software and Tools

Network virtualization allows hardware and software network resources as well as network functionality to be combined into a single, software-defined administrative unit which is called a virtual network. We distinguish external network virtualization that combines many networks into a unifying network, and internal network virtualization that provides network functionality to the processes and containers running on a single server.

- We will not cover this topic in this introductory class. However students can contribute a section or chapter

10.7 VIRTUAL MACHINE MANAGEMENT WITH QEMU



In this section we provide a short example on how to use QEMU. We will be starting with the installation, then create a virtual hard disk, install ubuntu on the disk and start the virtual machine. Next we will demonstrate how we can emulate a Raspberry Pi with QEMU. Lastly, we show how to use virsh.

10.7.1 Install QEMU

To install QEMU+KVM on Ubuntu/Linux Mint please use

```
$ sudo apt install qemu qemu-kvm libvirt-bin
```

On OSX QEMU can be installed with Homebrew

```
$ brew install qemu
```

10.7.2 Create a Virtual Hard Disk with QEMU

To create an image file with the size of 10GB and `qcow2` format (default format for QEMU images), run:

```
$ qemu-img create -f qcow2 testing-image.img 10G
```

Note that a new file called `testing-image.img` is now created at your home folder (or the place where you run the terminal). Note also that the size of this file is not 10 Gigabytes, it is around 150KB only; QEMU will not use any space unless needed by the virtual operating system, but it will set the maximum allowed space for that image to 10 Gigabytes only.

10.7.3 Install Ubuntu on the Virtual Hard Disk

Now that we have created our image file, if we have an ISO file for a Linux distribution or any other operating system and we want to test it using QEMU and use the new image file we created as a hard drive, we can run:

```
qemu-system-x86_64 \
-m 1024 \
-boot d \
-enable-kvm \
-smp 3 \
-net nic -net user \
-hda testing-image.img \
-cdrom ubuntu-16.04.iso
```

Explain the previous command part by part:

`-m 1024`:

Here we chose the RAM amount that we want to provide for QEMU when running the ISO file. We chose

1024MB here. You can change it if you like according to your needs.

`-boot -d:`

The boot option allows us to specify the boot order, which device should be booted first? -d means that the CD-ROM will be the first, then QEMU will boot normally to the hard drive image. We have used the `-cdrom` option as you can see at the end of the command. You can use `-c` if you want to boot the hard drive image first.

`-enable-kvm:`

This is a very important option. It allows us to use the KVM technology to emulate the architecture we want. Without it, QEMU will use software rendering which is very slow. That is why we must use this option, just make sure that the virtualization options are enabled from your computer BIOS.

`-smp 3:`

If we want to use more than 1 core for the emulated operating system, we can use this option. We chose to use 3 cores to run the virtual image which will make it faster. You should change this number according to your computer's CPU.

`-net nic -net user:`

By using these options, we will enable an Ethernet Internet connection to be available in the running virtual machine by default.

`-hda testing-image.img:`

Here we specified the path for the hard drive which will be used. In our case, it was the testing-image.img file

which we created before.

```
-cdrom ubuntu-16.04.iso:
```

Finally we told QEMU that we want to boot our ISO file *ubuntu-16.04.iso*.

10.7.4 Start Ubuntu with QEMU

Now, if you want to just boot from the image file without the ISO file (for example if you have finished installing and now you always want to boot the installed system), you can just remove the `-cdrom` option:

```
$ qemu-system-x86_64 -m 1024 -boot d -enable-kvm -smp 3 -net nic -net user -hda testing-ima
```

Please note QEMU *qemu-system-x86_64* emulates a 64-bit architecture.

10.7.5 Emulate Raspberry Pi with QEMU

First you have to download a pre-built kernel

```
$ wget https://raw.githubusercontent.com/dhruvvyas90/qemu-rpi-kernel/master/kernel-qemu-4.4
```

Next, you have to download the Raspbian image. Download a *.img* file from Raspbian website:

- <https://www.raspberrypi.org/downloads/raspbian/>

To start the emulator type in the following command to have QEMU emulate an ARM architecture:

```
$ qemu-system-arm -kernel ./kernel-qemu-4.4.34-jessie \
  -append "root=/dev/sda2 panic=1 rootfstype=ext4 rw" \
  -hda raspbian-stretch-lite.img \
  -cpu arm1176 -m 256 -machine versatilepb \
  -no-reboot -serial stdio
```

Pleaeee not that

- *kernel-qemu-4.4.34-jessie* is the pre-built kernel file.

- `raspbian-stretch-lite.img` is the Raspbian image file.

10.7.6 Resources

General

- Official website for `libvirt` is here: <https://libvirt.org/>
- Home page of KVM is here: https://www.linux-kvm.org/page/Main_Page
- QEMU home page: <https://www.qemu.org/>
- QEMU User Documentation: <https://qemu.weilnetz.de/doc/qemu-doc.html>
- Wikipedia page for QEMU: <https://en.wikipedia.org/wiki/QEMU>

Comparison

- <http://opensourceforu.com/2012/05/virtualisation-faceoff-qemu-virtualbox-vmware-player-parallels-workstation/>
- <https://stackoverflow.com/questions/43704856/what-is-the-difference-qemu-vs-virtualbox>

10.8 MANAGE VM GUESTS WITH VIRSH

`virsh` is a command line interface tool for managing guests and the hypervisor.

To initiate a hypervisor session with virsh :

```
virsh connect <name>
```

Where is the machine name of the hypervisor. If you want to initiate a read-only connection, append the above command with -readonly.

To display the guest list and their current states with virsh:

```
virsh list [ --inactive | --all]
```

The `-inactive` option lists inactive domains (domains that have been

defined but are not currently active). The -all domain lists all domains, whether active or not.

11 INFRASTRUCTURE AS A SERVICE



11.1 INTRODUCTION



❖ Learning Objectives

- Review IaaS service by prominent cloud providers.
- Understand how to write Python programs on managing virtual machines with libcloud.
- Understand how to write Python programs on managing data services with libcloud.
- Experiment with cloud providers while practically testing them.
⚠ be carfull with your allocation

[NIST](#) defines the term

Infrastructure as a Service (IaaS) as follows:

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

The key term is to provision fundamental computing resources. This means a user does not have to worry about managing the hardware allowing low level services such as the operating system or the network fabric to be the next higher interface for the user. The hardware fabric is managed by the cloud provider, while the

operating system level and their connectivity with each other is managed by the user.

We distinguish the following categories of infrastructure:

- compute resources
- network resources
- storage resources

We also like to remind you that such IaaS as parts of clouds can either be accessed public, private or in a hybrid fashion.

Within the next view section we will focus on some of the main providers of IaaS.

This includes

- Amazon Web Services
- Azure
- Google

Additionally, we also have

- Watson

which although provides IaaS focusses more on delivering AI platforms and related services to the community.

On the research side we will be focussing on

- Chameleon Cloud.

Some of the services listed provide a free small contingent of IaaS offerings that you can use. The use of this free tier will be sufficient to conduct this class.

A set of introductory slides is available at

 [Virtual Machines \(21\)](#)

11.2 AMAZON WEB SERVICES



11.2.1 AWS Products

Amazon Web Services offers a large number of products that are centered around their cloud services. These services have grown considerably over the years from the core offering related to virtual machine (EC2) and datastorage (S3). An overview of them is provided by Amazon in the following document:

- <https://d0.awsstatic.com/whitepapers/aws-overview.pdf>

We list the product in screenshots from their Product Web page panel in Figure below: [Figure 32](#), [Figure 33](#).

Compute	Networking & Content Delivery	Machine Learning	AR & VR
Amazon EC2	Amazon VPC	Amazon SageMaker	Amazon Sumerian
Amazon EC2 Auto Scaling	Amazon CloudFront	Amazon Comprehend	
Amazon Elastic Container Service	Amazon Route 53	Amazon Lex	
Amazon Elastic Container Service for Kubernetes	Amazon API Gateway	Amazon Polly	
Amazon Elastic Container Registry	AWS Direct Connect	Amazon Rekognition	
Amazon Lightsail	Elastic Load Balancing	Amazon Machine Learning	
AWS Batch		Amazon Translate	
AWS Elastic Beanstalk		Amazon Transcribe	
AWS Fargate		AWS DeepLens	
AWS Lambda		AWS Deep Learning AMIs	
AWS Serverless Application Repository		Apache MXNet on AWS	
Elastic Load Balancing		TensorFlow on AWS	
VMware Cloud on AWS			
Storage	Developer Tools	Analytics	Application Integration
Amazon Simple Storage Service (S3)	AWS CodeStar	Amazon Athena	Amazon MQ
Amazon Elastic Block Storage (EBS)	AWS CodeCommit	Amazon EMR	Amazon Simple Queue Service (SQS)
Amazon Elastic File System (EFS)	AWS CodeBuild	Amazon CloudSearch	Amazon Simple Notification Service (SNS)
Amazon Glacier	AWS CodeDeploy	Amazon ElasticSearch Service	AWS AppSync
AWS Storage Gateway	AWS CodePipeline	Amazon Kinesis	AWS Step Functions
AWS Snowball	AWS Cloud9	Amazon Redshift	
AWS Snowball Edge	AWS X-Ray	Amazon QuickSight	
AWS Snowmobile	AWS Tools & SDKs	AWS Data Pipeline	
		AWS Glue	
Database	Management Tools	Security, Identity & Compliance	Customer Engagement
Amazon Aurora	Amazon CloudWatch	AWS Identity and Access Management (IAM)	Amazon Connect
Amazon RDS	AWS Auto Scaling	Amazon Cloud Directory	Amazon Pinpoint
Amazon DynamoDB	AWS CloudFormation	Amazon Cognito	Amazon Simple Email Service (SES)
Amazon ElastiCache	AWS CloudTrail	Amazon GuardDuty	
Amazon Redshift	AWS Config	Amazon Inspector	
Amazon Neptune	AWS OpsWorks	Amazon Macie	
AWS Database Migration Service	AWS Service Catalog	AWS Certificate Manager	
	AWS Systems Manager	AWS CloudHSM	
Migration	AWS Trusted Advisor	AWS Directory Service	
AWS Migration Hub	AWS Personal Health Dashboard	AWS Key Management Service	
AWS Application Discovery Service	AWS Command Line Interface	AWS Organizations	
AWS Database Migration Service	AWS Management Console	AWS Single Sign-On	
AWS Server Migration Service	AWS Managed Services	AWS Shield	
AWS Snowball	Media Services		Business Productivity
AWS Snowball Edge	Amazon Elastic Transcoder		Alexa for Business
AWS Snowmobile	Amazon Kinesis Video Streams		Amazon Chime
	AWS Elemental MediaConvert		Amazon WorkDocs
	AWS Elemental MediaLive		Amazon WorkMail
	AWS Elemental MediaPackage		
	AWS Elemental MediaStore		
		Desktop & App Streaming	
		Amazon WorkSpaces	
		Amazon AppStream 2.0	
		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	
		Game Development	
		Amazon GameLift	
		Amazon Lumberyard	
		Software	
		AWS Marketplace	
		AWS Cost Management	
		AWS Cost Explorer	
		AWS Budgets	
		Reserved Instance Reporting	
		AWS Cost and Usage Report	

Figure 32: AWS Products 1

AWS Snowball	AWS Elemental MediaTailor	AWS WAF
AWS Snowball Edge		AWS Artifact
AWS Snowmobile		
		Mobile Services
		AWS Mobile Hub
		Amazon API Gateway
		Amazon Pinpoint
		AWS AppSync
		AWS Device Farm
		AWS Mobile SDK
		AWS Cost Management
		AWS Cost Explorer
		AWS Budgets
		Reserved Instance Reporting
		AWS Cost and Usage Report

Figure 33: AWS Products 2

Service offerings are grouped by categories:

- Compute
- Storage
- Database
- Migration
- Networking and Content Delivery
- Developer Tools
- Management Tools
- Media Services

- Machine Learning
- Analytics
- Security and Identity Compliance
- Mobile Services
- AR and VR
- Application Integration
- Customer Engagement
- Business Productivity
- Desktop and App Streaming
- Internet of Things
- Game Development
- Software
- Aws Core Management

Within each category you have several products. When choosing products from AWS it is best to start with the overview paper and identify products that can be of benefit to you. For our purpose we focus on the traditional Compute and Storage offerings. We list the products that are available In Sep. 2018 next.

11.2.1.1 Virtual Machine Infrastructure as a Services

Amazon offers a large number of services related to virtual machine management

- [Amazon EC2](#)
- [Amazon EC2 Auto Scaling](#)

11.2.1.2 Container Infrastructure as a Service

Amazon offers the following container based services

- [Amazon Elastic Container Service](#)
- [Amazon Elastic Container Service for Kubernetes](#)
- [Amazon Elastic Container Registry](#)

11.2.1.3 Serverless Compute using AWS Lambda

In addition to these services a number of additional compute services Serverless computing or FaaS is a new cloud computing paradigm that has gained popularity recently. Serverless Computing with AWS Lambda. Serverless computing or Function as a Service (FaaS) is a new cloud computing paradigm that has gained popularity recently. AWS Lambda was one of the first serverless computing services that was made available to the public, Serverless computing allows users to run small functions in the cloud without having to worry about resource requirements. More information regarding AWS Lambda can be found in the following document

- <https://aws.amazon.com/lambda/>

11.2.1.4 Serverless Compute using AWS Lambda

Serverless computing or FaaS is a new cloud computing paradigm that has gained popularity recently.

11.2.1.5 Storage

AWS provides many storage services that users can leverage for developing applications and solutions. The list below showcases AWS storage services. Amazon offers the following storage services

- [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](#)

11.2.1.6 Databases

AWS also provides many data base solutions. AWS has both SQL based databases and NoSQL based databases. The list below shows the database services that AWS offers. And other database related services

- [Amazon Aurora](#)
- [Amazon RDS](#)
- [Amazon DynamoDB](#)
- [Amazon ElastiCache](#)
- [Amazon Redshift](#)
- [Amazon Neptune](#)
- [AWS Database Migration Service](#)
- [AWS Marketplace](#)

11.2.2 Locations

As the following figure shows: [Figure 34](#).

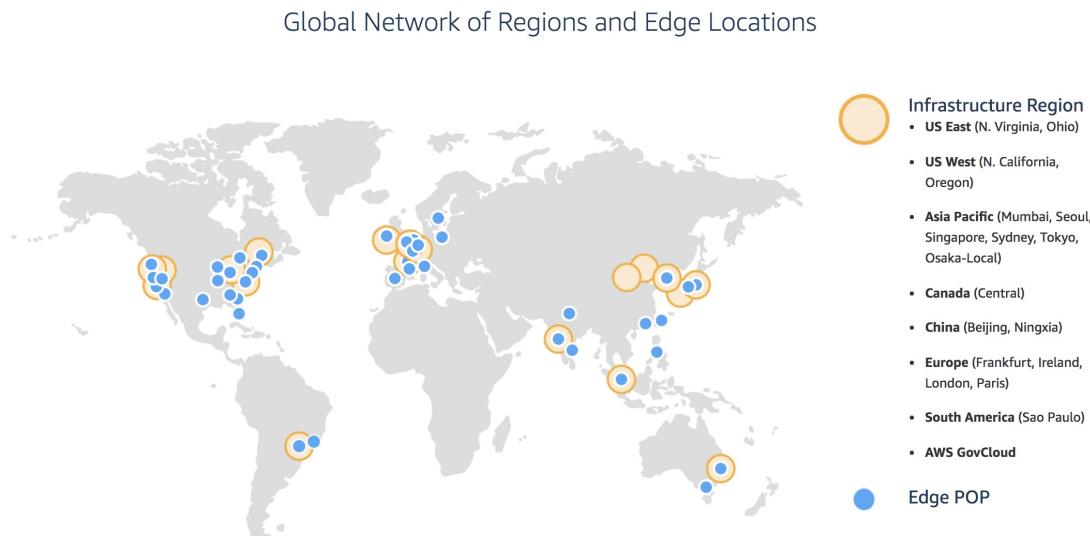


Figure 34: AWS-Locations

11.2.3 Creating an account

In order to create a AWS account you will need the following

- A valid email address
- A credit/debit card
- A valid phone number

First you need to visit the AWS [signup page](#) and click "Create Free Account". You will then be asked to provide some basic details including your email address as shown in the image below: [Figure 35](#).

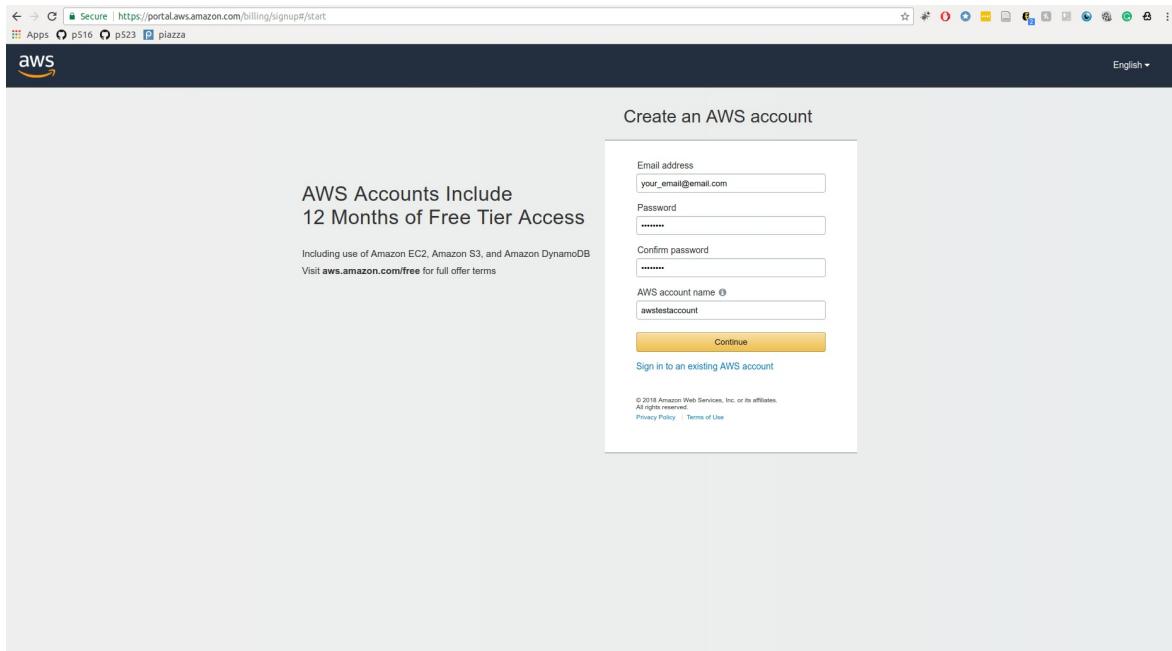


Figure 35: AWS Signup

Next you will be asked to provide further details such as your name, address and phone number. After the additional details have been provided. AWS will ask for credit/debit card details as shown below: [Figure 36](#). They require this information to verify your identity and make sure they have a method to charge you if needed. However no charges will be applied to your credit/debit card unless you use the AWS services and exceed the free tier limits, which will be discussed in the next section.

The screenshot shows a web browser window with the URL <https://portal.aws.amazon.com/billing/signup#/paymentInformation>. The browser's address bar and various icons are visible at the top. The main content area is titled "Payment Information". It contains fields for "Credit/Debit card number", "Expiration date" (set to 08/2018), "Cardholder's name", and "Billing address". The billing address is pre-filled with "2805 E 10th St Smith Research Center Bloomington IN 47408 US". There is also a radio button option for "Use a new address". At the bottom right is a yellow "Secure Submit" button. A small note at the bottom left states "Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the [AWS Free Tier Limits](#). Review [frequently asked questions](#) for more information." The AWS logo is in the top left corner of the page.

Figure 36: Payment Information

After the credit/debit card has been verified AWS will use your phone number to verify your identity. Once you are logged into your account you will be able to sign into the console, from the link on the top right corner in your account. Once you are in the AWS console the services tab in the left top corner will allow you to access all the services that are available to you through AWS as shown in the image below: [Figure 37](#).

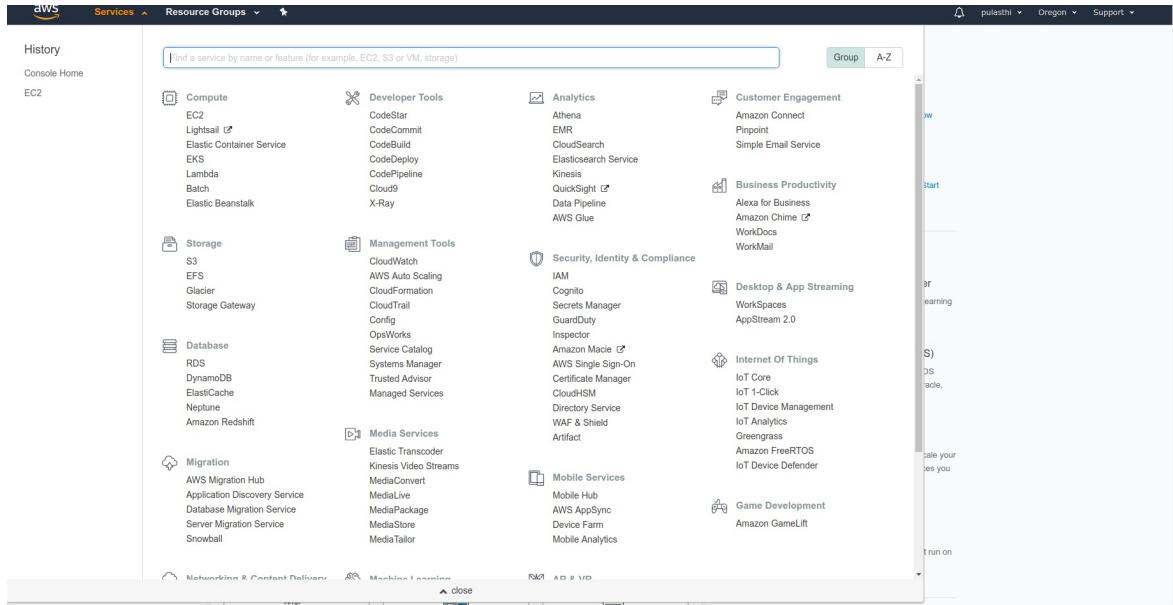


Figure 37: AWS Console

11.2.4 AWS Command Line Interface

11.2.4.1 Introduction

Amazon's CLI allows for programmatic interaction with AWS product through the command line. CLI provide many pre-built functions that allow for interaction with Amazon's Elastic Compute Cloud (EC2) instances and S3 storage.

11.2.4.2 Prerequisites

- [Linux](#)
- [Python](#)
- [PIP](#)
- [AWS Account](#)
- [AWS Key Pair](#)

11.2.4.2.1 Install CLI

Run the following code to install CLI.

```
pip install awscli
```

11.2.4.2.2 Configure CLI

Using the following code to configure AWS using. You will need to specify four parameters:

1. AWS Access Key ID
2. AWS Secret Access Key
3. Default region name (this is the default region that will be used when you create EC2 instances)
4. Default output format (the default format is json)

```
aws configure
```

11.2.5 AWS Admin Access

11.2.5.1 Introduction

In order to access various AWS functionality remotely (through command-line) you must enable administrative access.

11.2.5.2 Prerequisites

- [Set up AWS account](#)
- [Install and configure AWS CLI](#)
- [Linux environment](#)
- [AWS Key Pair](#)

11.2.5.3 Setting up admin access using AWS CLI

11.2.5.3.1 Create an admin security group

```
aws iam create-group --group-name Admins
```

11.2.5.3.2 Assign a security policy to the created group granting full admin access

```
aws iam attach-group-policy --group-name Admins --policy-arn arn:aws:iam::aws:policy/Admin:
```



11.2.6 Understanding the free tier

AWS provides a set of services free of charge. These free services are to allow new users test and experiment with various AWS services without worrying about any cost. Free services are provided as a product that is free until a certain amount of usage, that is if you exceed those limits you will be charged for the additional usage. However the free quotas are typically more than sufficient for testing and learning purposes. For example under the free tier you are able to use 750 hours of EC2 resources per month for the first 12 months after account creation. However it is important to make note of important details that are included in the limits. For example for the 750 hours of free EC2 usage, you can only use "EC2 Micro" instances, using any other instance type for your EC2 machine will not fall under the free tier agreement and you will be charged for them, see picture below: [Figure 38](#). To view all the AWS free tier details visit [AWS Free Tier](#)

<p>12 months free and always free products</p> <p>AWS Free Tier includes offers that expire 12 months following sign up and others that never expire.</p> <p>Learn more »</p>	<p>COMPUTE</p> <p>Amazon EC2</p> <p>750 Hours</p> <p>per month</p> <p>Resizable compute capacity in the Cloud</p> <p>Learn more about Amazon EC2 »</p> <p><small>EXPAND DETAILS ^</small></p>	<p>ANALYTICS</p> <p>Amazon QuickSight</p> <p>1 GB</p> <p>of SPICE capacity</p> <p>Fast, easy-to-use, cloud-powered business analytics service at 1/10th the cost of traditional BI solutions</p> <p>Learn more about Amazon QuickSight »</p> <p><small>EXPAND DETAILS ^</small></p>
<p>DATABASE</p> <p>Amazon RDS</p> <p>750 Hours</p> <p>per month of db.t2.micro database usage (applicable DB engines)</p> <p>Managed Relational Database Service for MySQL, PostgreSQL, MariaDB, Oracle BYOL, or SQL Server</p> <p>Learn more about Amazon RDS »</p> <p><small>EXPAND DETAILS ^</small></p>	<p>STORAGE & CONTENT DELIVERY</p> <p>Amazon S3</p> <p>5 GB</p> <p>of standard storage</p> <p>Secure, durable, and scalable object storage infrastructure</p> <p>Learn more about Amazon S3 »</p> <p><small>EXPAND DETAILS ^</small></p>	<p>COMPUTE</p> <p>AWS Lambda</p> <p>1 Million</p> <p>free requests per month</p> <p>Compute service that runs your code in response to events and automatically manages the compute resources</p> <p>Learn more about AWS Lambda »</p> <p><small>EXPAND DETAILS ^</small></p>

Figure 38: Free tier

Basically there are two categories in the free tier,

- 12 months free
- Always free

12 months free offer are only good for the first 12 months after you create your AWS account. The always free offer are valid even after the first 12 months.

11.2.7 Important Notes

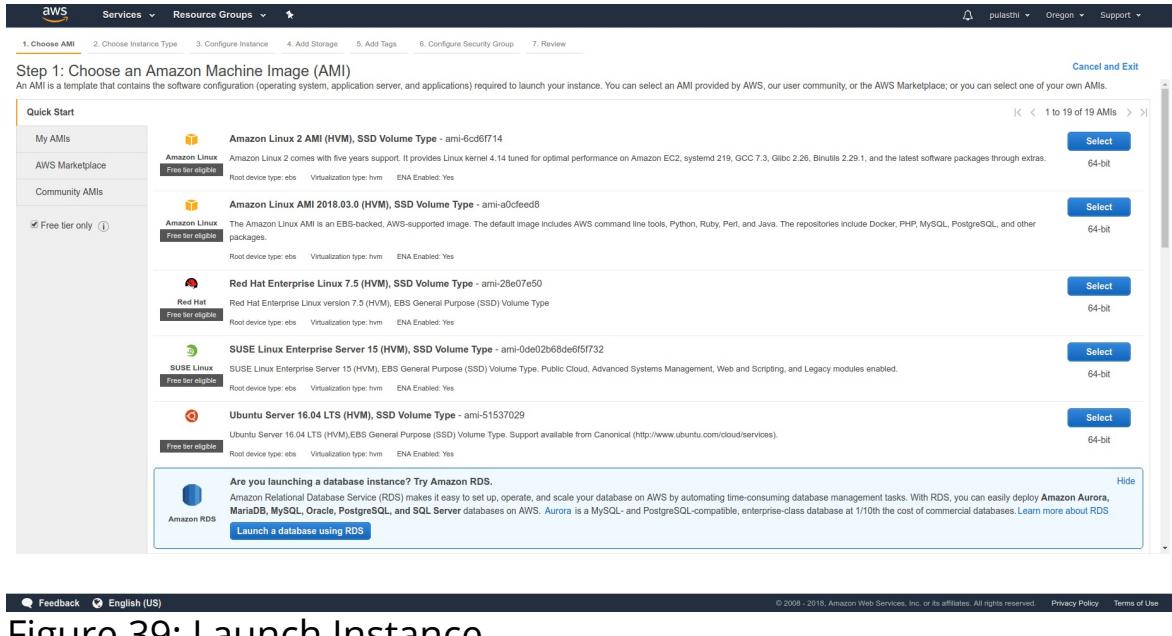
⚠ When using AWS services make sure you understand how and when you will be charged for. For example if you are using an EC2 to run some application, usage of the instance will be calculated from the time you started the instance to the time you stop or terminate the instance. So even if you do not use the application itself, if you are have the instance in an active mode that will be added to the usage hours and you will be billed if you exceed the 750 hour limit. In EC2 even if you stop the instance you might be charged for data that is stored in the instance so terminating it would be the most safest option if you do not have any important data stored in the instance. You can look up other such tricky scenarios at [Avoiding Unexpected Charges](#) to make sure you will not get an unexpected bill

11.2.8 Introduction to the AWS console

As we discussed previously we can access all the service and product offerings that are provided by AWS from the AWS console. In the following section we will look into how we can start and stop a virtual machine using AWS EC2 service. Please keep in mind that this will reduce time from your free tier limit of 750 hours/month, So be careful when starting EC2 instances and make sure to terminate them after you are done.

11.2.8.1 Starting a VM

To go to the EC2 services you can click on the services link on the top left corner in the console and then click on EC2 which is listed under “Compute”. Then you will see a blue button labeled “Launch instance”. Click on the button and the console will take you to the page shown below: [Figure 39](#). Notice that the check box for “Free tier only” is clicked to make sure the instance type we choose is eligible for the free tier hours. The instance type you select defines the properties of the virtual machine you are starting such as RAM, Storage, processing power. However since we are using instances that are free tier eligible we will only be able to use “EC2 Micro instances”. You can also select an OS that you want to be started as the virtual machine. We will select “Ubuntu Server 16.04 LTS” as our Operating system. press the blue select button to do so.



Feedback English (US)

Figure 39: Launch Instance

Once you select the OS type you will be asked to select the instance type. You can notice that only the “t2.micro” is marked as free tier eligible as shown in the image below: [Figure 40](#). Now that you have selected all the basic details press the “Review and Launch” button located in the button right corner. This will give you a summary of your current selections.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	i2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
General purpose	i2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	i2.medium	2	4	EBS only	-	Low to Moderate	Yes

Figure 40: Instance Type

11.2.8.1.1 Setting up key pair

Before we can launch the VM we need to perform one more step. We need to setup a SSH key pair for the new VM. Creating this will allow us to access our VM through SSH. Once you click on the launch button, you will get the following dialog box: [Figure 41](#). If you already

have worked with SSH keys and if you already have a key pair you can use it, otherwise you can create a new key pair as we will do. To create a new key pair select the “Create a new key pair” in the first drop down box and enter a name of your choosing as the name. Next you need to download and save the private key, Keep the private key in a safe place and do not delete it since you will need it when you are accessing the VM (This tutorial will not cover accessing the VM through SSH but you need to keep the private key so you can use the same key value pair later). Once you have downloaded the private key, the “Launch Instance” button will activate. Press this button to start the VM.

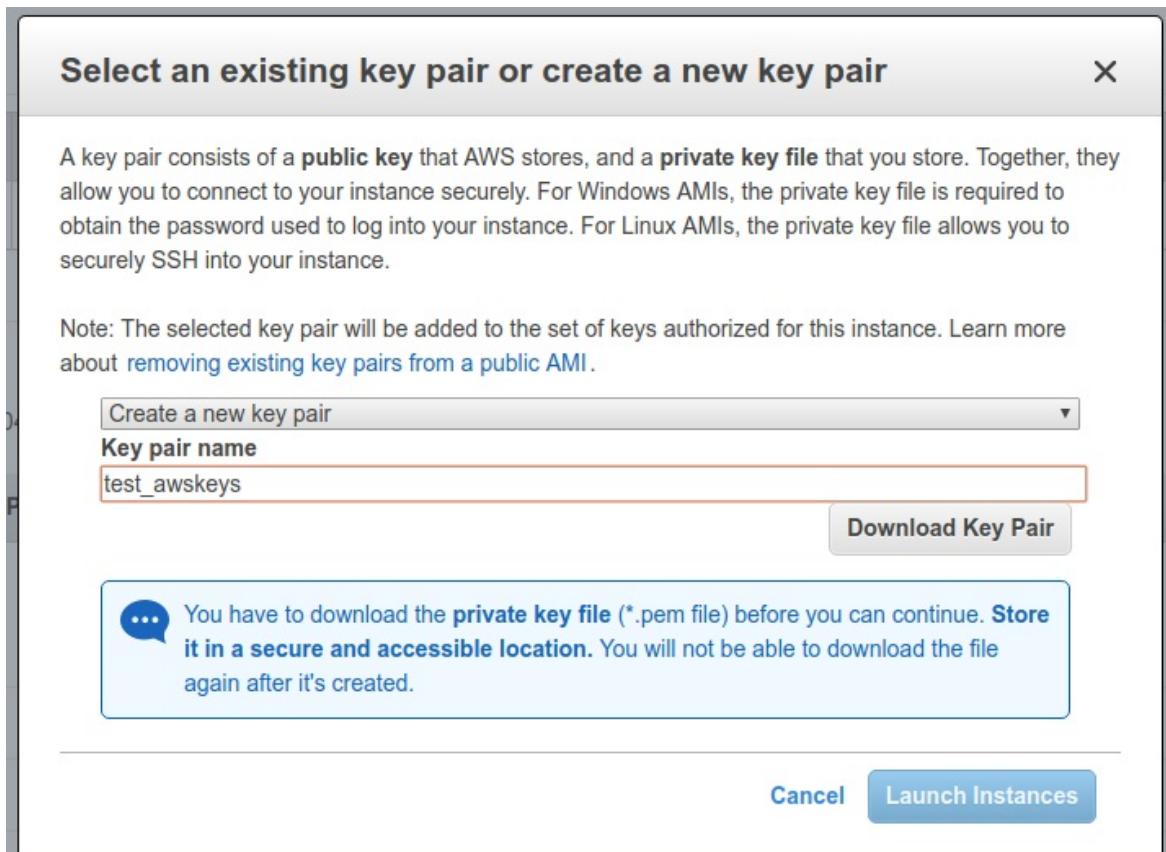


Figure 41: Key Pair

After starting the instance go back to the EC2 dashboard (Services -> EC2). Now the dashboard will show the number of running instance as shown in the image below: [Figure 42](#). If you do not see it initially, refresh the page after a little while, starting the VM may take a little time so the dashboard will not be updated until the VM starts.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (Oregon) region

Service Health		Scheduled Events	
Service Status:	US West (Oregon):	Scheduled Events	
<input checked="" type="radio"/> US West (Oregon):	No events		
Availability Zone Status:			
<input checked="" type="radio"/> us-west-2a: Availability zone is operating normally			
<input checked="" type="radio"/> us-west-2b: Availability zone is operating normally			
<input checked="" type="radio"/> us-west-2c: Availability zone is operating normally			

[Service Health Dashboard](#)

Figure 42: Running Instance1

Now to get a more detailed view click on the “Running Instances” link. This will give you the following view: [Figure 43](#). Is shows the current instance that you are running

Launch Instance												
Actions												
<input type="text"/> Filter by tags or attributes or search by keyword												
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring	
i-0e4530cbf4f5f9ce	i-0e4530cbf4f5f9ce	t2.micro	us-west-2a	running	Initializing	None	ec2-54-201-224-79.us...	54.201.224.79		test_awakekeys	disabled	

Figure 43: Running Instance2

11.2.8.2 Stopping a VM

In AWS EC2 you can either stop a VM or terminate it. If you terminate it you will lose all the data that was stored in the VM as well, simply stopping will save the data for future use if you restart the instance again. In order to stop the VM you can select the VM machines you want to stop from the GUI and go to “Actions -> Instance status” and click on stop: [Figure 44](#). This will stop your VM machine.

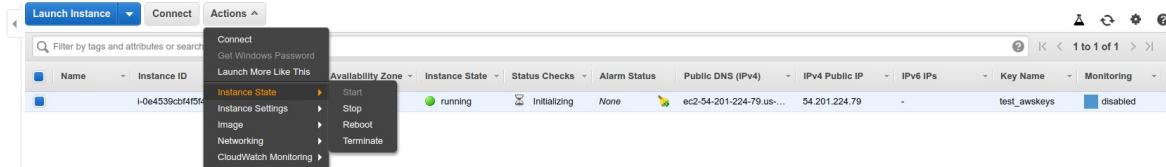


Figure 44: Instance Stop

After a little while the dashboard will show the instance as stopped as the following: [Figure 45](#). If you want to go further and terminate the instance you can again go to “Actions -> Instance status” and select terminate, which will terminate the VM.

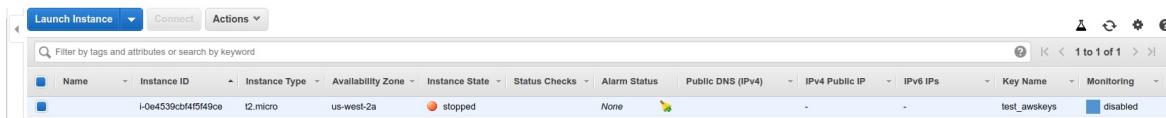


Figure 45: Stopped Instance

11.2.9 Access from the Command Line

AWS also provides an command line interface that can be used to manage all the AWS services through simple commands. below are two example commands.

```
aws s3 <Command> [<Arg> ...]
aws ec2 <Command> [<Arg> ...]
```

You can find more information regarding the AWS CLI in the following documents.

- AWS Command Line: <https://aws.amazon.com/cli/>
- AWS Command Line reference: <https://docs.aws.amazon.com/cli/latest/reference/>
- EC2: <https://docs.aws.amazon.com/cli/latest/reference/ec2/index.html>
- S3: <https://docs.aws.amazon.com/cli/latest/reference/s3/index.html>

Amazon Web Services (AWS) is a cloud platform that provides a large number os services for individuals and enterprises. You can get an

overview of the AWS offering at [Amazon Web Services Overview](#). This section will guide through the processes of creating an AWS account and explain the free tier details so that you can leverage the tools and products available in AWS for your work and research.

11.2.10 Access from Python

11.2.11 Boto

Boto is a Python software development kit specifically targeting Amazon Web Services (AWS). It allows access to services such as S3 and EC2. It is using object oriented programming paradigms to access the lower level services. The advantage is that it is written just for Amazon and thus we assume it will be developed with high quality due to its specialization. However this is also its limitation as in contrast to libcloud it does not support other cloud providers. Hence it bears the risk of vendor lockin. Boto is maintained in github.

Documentation about boto can be found at

- <https://boto3.readthedocs.io/en/latest/>
- <https://github.com/boto/boto3>

11.2.12 libcloud

“Libcloud is a Python library for interacting with many of the popular cloud service providers using a unified API. It was created to make it easy for developers to build products that work between any of the services that it supports.” A more detailed description on Libcloud and how you can use it to connect with AWS is provided in the Section [Python libcloud](#).

For more information about the features and supported providers, please refer to the [documentation](#)

11.3 MICROSOFT AZURE



Microsoft Azure is a cloud computing service created by Microsoft. It includes computing services and products for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

11.3.1 Products

Just as Microsoft offers a large number of services. We included the from Microsoft in Sep. 2018 highlighted services in the appendix, with convenient links to them.

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

We focus next on the

- compute,
- container, and
- data resources

For a more elaborate list please consult our Appendix. To see the complete list lease visit the Microsoft Web page via this [link](#).

11.3.1.1 Virtual Machine Infrastructure as a Services

Source: <https://support.microsoft.com/en-us/allproducts>

Microsoft offers core IaaS [Compute](#) compute resources. This includes the following services:

- [Virtual Machines](#) to provision Windows and Linux virtual machines
- [Virtual Machine Scale Sets](#) to manage and scale thousands of Linux and Windows virtual machines

11.3.1.2 Container Infrastructure as a Service

Microsoft offers [Containers](#) to allow for the development of containerized applications. This includes:

- [Azure Kubernetes Service \(AKS\)](#) to provide access to Kubernetes as a Service so that deployment, management, and operations of Kubernetes can be conducted on the cloud resources offered.
- [Service Fabric](#) to develop microservices and orchestrate containers on Windows or Linux as part of the infrastructure
- [Container Instances](#) to run containers on Azure without managing servers which seems unrelated to kubernetes
- [Container Registry](#) to store and manage container images for deployments

11.3.1.3 Databases

Storage is offered through a variety of [Database](#) services to provide access to enterprise-grade, and fully managed services.

- [Azure Cosmos DB](#) is a globally distributed, multi-model database for any scale
- [Azure SQL Database](#) is a managed relational SQL database as a service
- [Azure Database for MySQL](#) is a managed MySQL database as a service
- [Azure Database for PostgreSQL](#) is a managed PostgreSQL database service
- [SQL Server on Virtual Machines](#) allowing to host enterprise SQL Server apps in the cloud
- [SQL Data Warehouse](#) is an elastic data warehouse as a service with enterprise-class features
- [Azure Database Migration Service](#) simplifies on-premises database migration to the cloud
- [Redis Cache](#) which provides a Redis Cache as a service to support high-throughput and low-latency data access
- [SQL Server Stretch Database](#) which supports dynamically stretch on-premises SQL server databases to Azure

11.3.1.4 Networking

We will not go much into the network offerings at this time

11.3.2 Registration

In order for you to register in Azure and start your free account, you will need to go to

- <https://azure.microsoft.com/en-us/free/>

You will see an image such as below: [Figure 46](#).

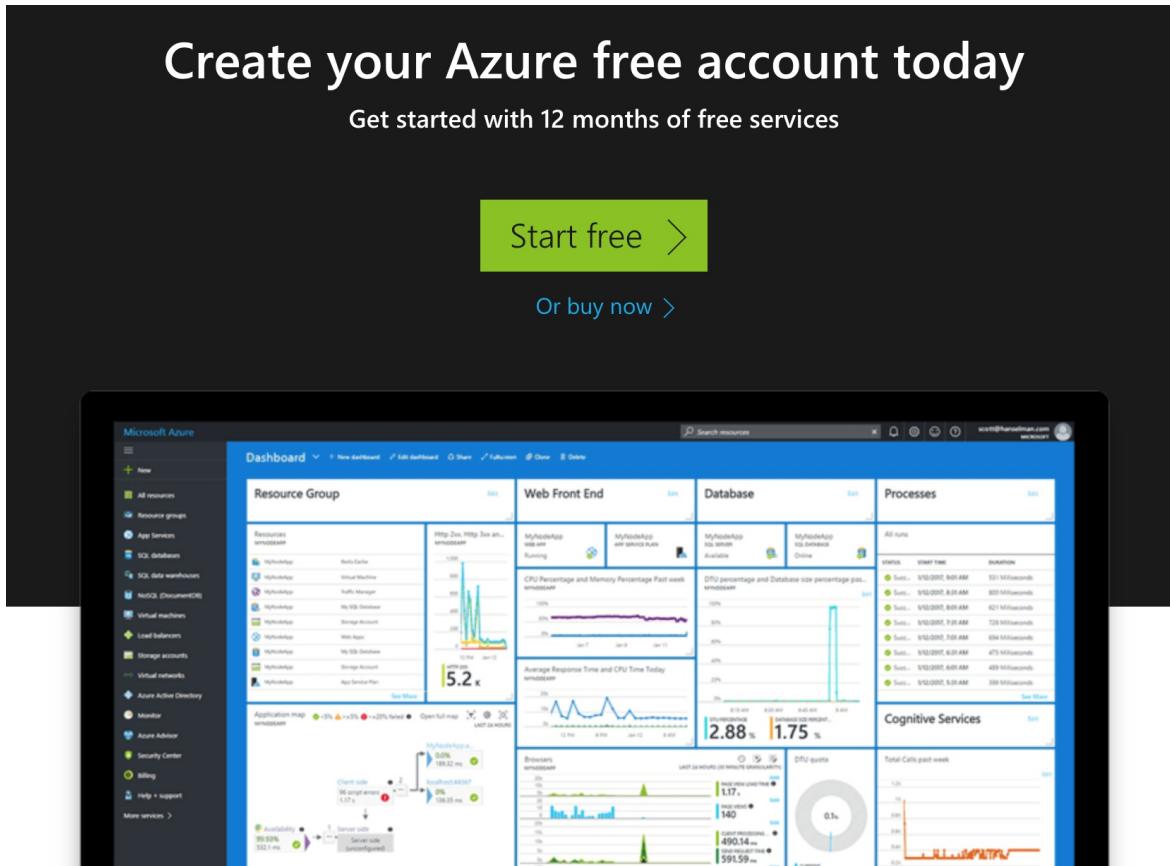


Figure 46: Registration

On that image you click the `start free` button to obtain a free one year account. You will have to either create a new Microsoft account or use the one from Indiana University which will be your IU id followed by the `???` domain. You will be redirected to the single sign on from IU to proceed. If you use another e-mail you can certainly do that and your free account is not associated with the IU account. This could be your Skype account or some other e-mail. After registration you will be provided with 12 months of free usage of a few selected services and \$200 credits for 30 days.

The services that you have access to include:

- Linux Virtual Machines (750 Hours)
- Windows Virtual Machines (750 Hours)
- Managed Disks (64 GB X 2)
- Blob Storage (5 GB)

- File Storage (5 GB)
- SQL Database (250 GB)
- Azure Cosmos DB (5 GB)
- Bandwidth (Data Transfer 15 GB)
- In case Azure changes the product list, please refer to the official page for a full list of free products:
<https://azure.microsoft.com/en-us/free/>

11.3.3 Introduction to the Azure Portal

Azure can be accessed via a portal. An introductory video from Microsoft provides you with some elementary information:

 [Introduction to Azure Portal](#)

11.3.4 Creating a VM

Choose `Create a resource` in the upper left-hand corner of the Azure portal. Select a VM name, and the disk type as SSD, then provide a username. The password must be at least 12 characters long and meet the defined complexity requirements. As the following: [Figure 47.](#)

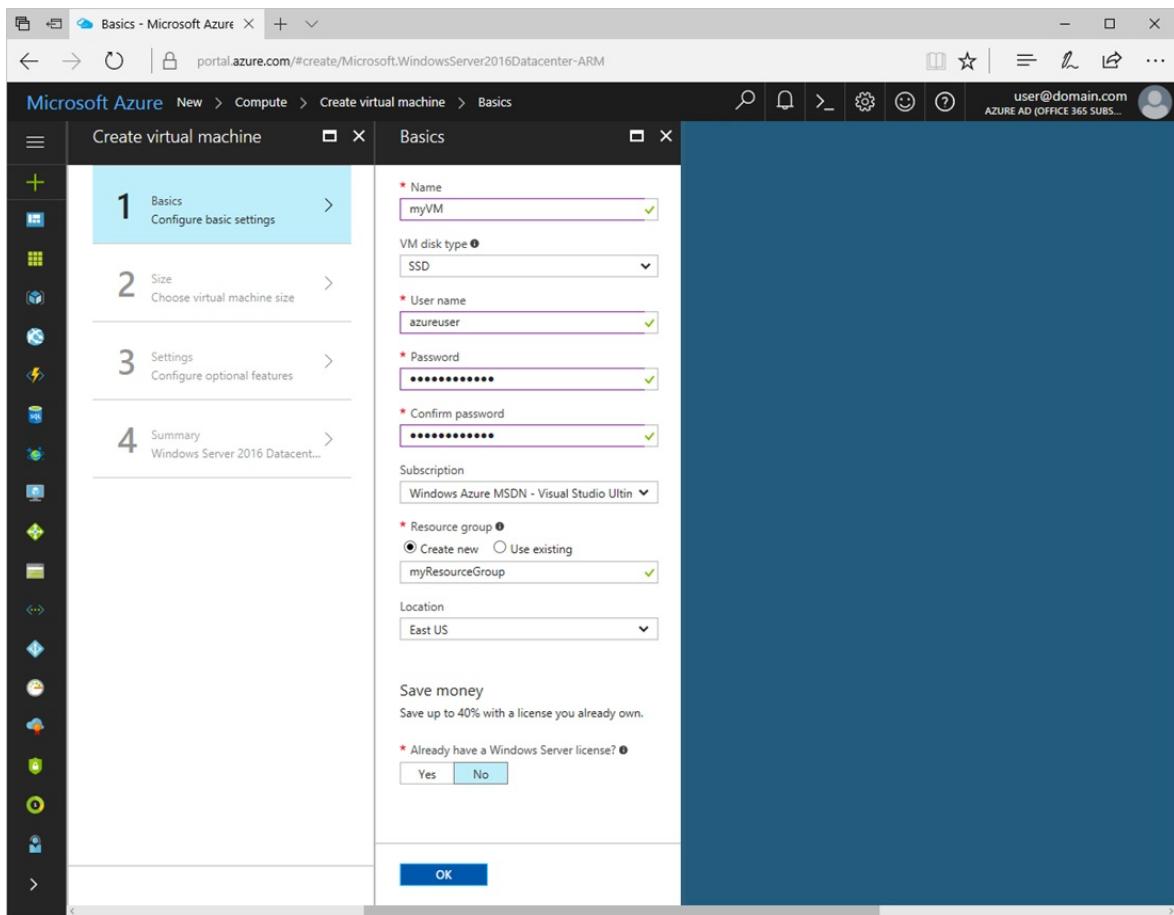


Figure 47: Creating a VM

Source: <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/media/quick-create-portal/create-windows-vm-portal-basic-blade.png>

11.3.5 Starting a VM

Now we like to introduce you how to start a VM. Please note that VMS do cost and reduce your free hours on Azure. Hence you need to make sure you carefully review the charging rates and chose VM sizes and types that minimize your charges.

A VM can be started through the Portal as follows: [Figure 48](#).

- On the overview tab, a VM can be started by clicking the `start` button.

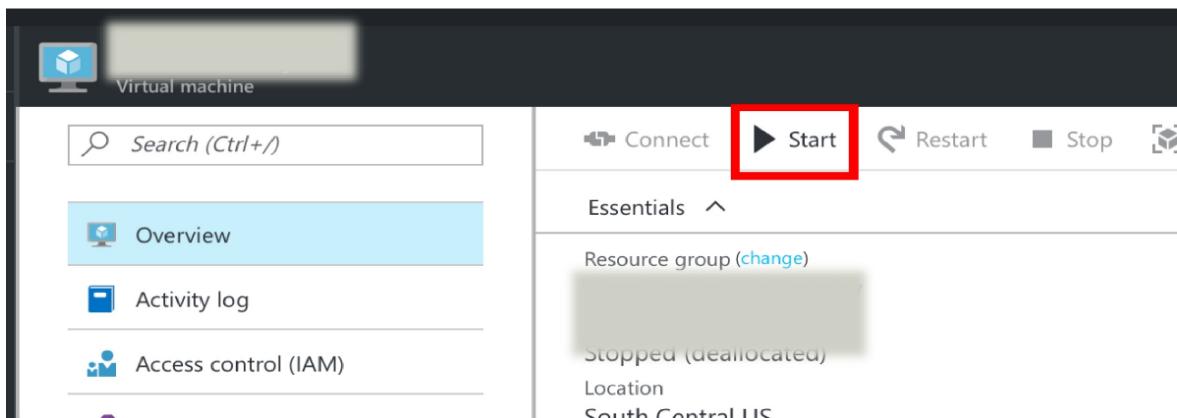


Figure 48: Start button

11.3.6 Stopping the VM

It is the most important to stop your VMS once they are not in used, or you get continuously charged. The portal allows you to see the list of VM that you run as follows

To shut a VM down, please do the following: see [Figure 49](#).

- On the overview tab, a VM can be started by clicking the `Stop` button.

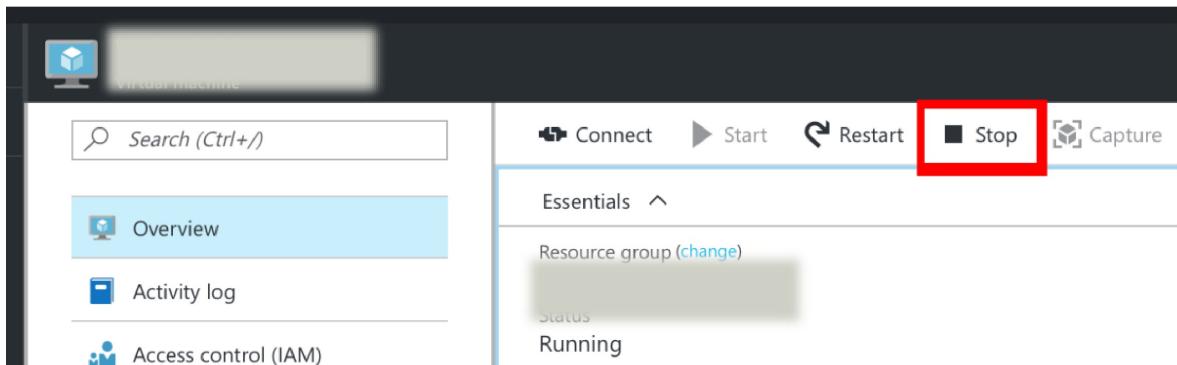


Figure 49: Stop button

11.3.7 Exercises

E.Azure.0:

Identify all products related to IaaS service offerings and mark them with ★ 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.

E.Azure.1:

What is the difference between terminating, shutting down, and suspension?

E.Azure.2:

Do I get charged when the VM is suspended, terminated, shutdown?

E.Azure.3:

How do I resume a VM if it is suspended?

11.4 WHAT IS IBM WATSON AND WHY IS IT IMPORTANT?

In years past we traditionally typed our questions into a query based search engine and it would return relevant content. As we start interacting with our devices more in conversation through natural language processing, we need an answer to a question - not a list of relevant information. We need the power of question answering (QA) technology.

IBM's Watson's is well known for its ability to play and successfully win the popular gameshow, jeopardy! IBM startled the world in 2011 when Watson beat Jeopardy! pros Ken Jennings and Brad Rutter over several rounds. Watson completed the formidable task by combining 15 terabytes of human knowledge with a variety of computer disciplines including automated reasoning, natural language processing, knowledge representation, information retrieval and machine learning.

IBM's goal of having computers understand the questions humans ask while providing answers in a similar fashion is not unique to them. In recent years, products like Amazon's Alexa and Google Home have brought the awareness of this capability mainstream for millions of households. In short, it has become a race to serve up relevant content with the least amount of effort in the most consumable format.

11.4.1 How can we use Watson?

IBM's Watson has a rich set of Developer Services (<https://www.ibm.com/watson/developer/>) that allow users to stand on the shoulders of the IBM developers using their AI framework to "bolt on" new or improved applications that sit on top.

There are a breadth of services available. Watson Discovery is used to mine through data to find trends and surface patterns. Watson Visual Recognition is used to classify content using machine learning. Watson Assistant provides a framework for chatbots and virtual agents.

While The next section walks through how to create a free account, let's continue with an example of leveraging a foundation and building on-top with Watson Assistant Basic. Please see steps: [Figure 50](#), [Figure 51](#), [Figure 52](#), [Figure 53](#), [Figure 54](#), [Figure 55](#)

Instead of starting with a blank page IBM steps are put in place and working examples can be customized.

The screenshot shows the IBM Cloud interface for the Watson Assistant Basic Starter Kit. At the top, there's a navigation bar with 'IBM Cloud' and links for 'Catalog', 'Docs', 'Support', and 'Manage'. Below the navigation, the title 'Watson Assistant Basic' is displayed with a speech bubble icon. Underneath the title, it says 'Web App • Lite'. The main content area has a heading 'Overview' followed by a detailed description of the starter kit. A section titled 'This starter kit will help you' lists three bullet points: 'Available in Node.js, Python and Java', 'Deploy Locally and to the IBM Cloud', and 'Access to Watson Assistant (formerly Conversation)'. At the bottom of this section, there's a large button labeled 'Ready to get started?'.

Figure 50: Step 1

This screenshot shows the 'App Details' page for the 'Watson Assistant Basic ASKOI' application. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', 'Manage', and a search bar. The main content area displays the app's name, 'Watson Assistant Basic ASKOI', and its status as 'Node.js'. It features sections for 'App Details', 'Resources (1)', 'Deploy your App', 'Credentials', and 'Knowledge Guide'. The 'Resources (1)' section shows a single resource named 'Watson Assistant (formerly Conversation)'. The 'Deploy your App' section contains a 'Deploy to Cloud' button. The 'Credentials' section shows a JSON configuration file:

```
{
  "conversation": [
    {
      "name": "watson-assistant-bas-conversation-1548449029875",
      "credentials": [
        {
          "url": "https://gateway.wdc.watsonplatform.net/assistant/api"
        }
      ]
    }
  ]
}
```

The 'Knowledge Guide' section provides step-by-step instructions for getting started:

- Click "Download Code" to download a .zip file of your generated app
- Extract the .zip file
- Install Node.js (LTS supported versions)
- Open a terminal and go to your starter kit directory
- Install the dependencies: npm install
- Start the application: npm start
- Go to //localhost:3000 in a web browser to view the application
- Click on "Launch tool" to train the Assistant service

Figure 51: Step 2

The screenshot shows the IBM Watson Assistant landing page. At the top, there are navigation links for 'Home' and 'Workspaces'. Below this, a section titled 'Introducing IBM Watson Assistant' features a sub-section 'Watson Conversation is evolving to simplify how you build and scale virtual assistants. See what's new'. To the right, there is a graphic of two speech bubbles. The main content area is titled 'Three easy steps' and lists three steps: 1. Create intents and entities, 2. Build your dialog, and 3. Test your dialog. Each step has a brief description and a 'Learn more' link. A large blue button at the bottom says 'Get started now'.

Figure 52: Step 3

The screenshot shows the 'Workspaces' page of the IBM Watson Assistant interface. The top navigation bar includes 'IBM Watson Assistant', 'Home', and 'Workspaces'. A tooltip message 'Info: Creating workspace from sample "Customer Service - Sample"' is displayed. The main area is titled 'Workspaces' with an upward arrow icon. On the left, there is a dashed box containing a 'Create a new workspace' button and a note about workspaces enabling separate intents, user examples, entities, and dialogs. It also mentions 'You are using 1 of 5 available workspaces in this instance.' and a 'Create' button with a plus sign. On the right, a workspace named 'Customer Service - Sample' is listed, described as 'A virtual assistant for customer service sample' in English (U.S.). It includes a 'Get started' button and a note that it was 'Last modified: just now'.

Figure 53: Step 4

#Customer_Care_Appointments

Intent name: #Customer_Care_Appointments

Description: Schedule or manage an in-store appointment.

Add user examples: Is there a cancellation fee

User examples (19) ▾

User Example	Added
are you available on tuesday	a minute ago
Can I book an in person session	a minute ago
can i book for tonight	a minute ago
can i make an appointment	a minute ago
can you make an appointment for me	a minute ago
Could I speak to someone in the store next tuesday?	a minute ago

Figure 54: Step 5

Customize "Small Talk"

Digressions

This folder has **edited** digressions settings ⓘ

Allow digressions into this folder

Users can digress to this folder from other dialog flows.

Return after digression

After the digression has been handled by a dialog flow within this folder, return to the dialog flow that was previously in progress.

Cancel Apply

Figure 55: Step 6

In the case above when I was trying the Watson Assistant It was not personal when I asked the Assistant's name so it can be modified to digress!

In addition to using these modules to help build there is also a variety

of APIs and services that can be used:

The list of APIs and services include: * Watson Assistant * Watson Discovery * Natural Language Understanding * Discovery News * Knowledge Studio * Language Translator * Natural Language Classifier * Personality Insights * Tone Analyzer * Visual Recognition * Speech to Text * Text to Speech

11.4.2 Creating an account

This section will guide through the processes of creating an IBM Watson account and explain the free tier details so that you can leverage the tools and products available in AWS for your work and research.

- A valid email address

First you need to visit the [IBM Watson home page](#) and click in the “Get Started Free” link on the top right corner. You will then be asked to provide some basic details including your email address as shown in the image [Figure 56](#).

Sign up for an IBMid and create your IBM Cloud account

Build on IBM Cloud for free with no time restrictions

Guaranteed free development with Lite plans
Develop worry-free and at no cost with cap based Lite plan services for as long as you like.

Start on your projects right away
Skip entering your credit card info and get working in just a few short steps.

Get a \$200 credit when you upgrade
After you upgrade to a Pay-As-You-Go account, you can use the credit to try new services or scale your projects. The credit is valid for 1 month and can be used with any of our IBM Cloud offerings.

Ready to get started? Sign up today!

Email*
your_email@email.com

First Name*

Last Name*

Country or Region*
United States

Password*

IBM may use my contact data to keep me informed of products, services and offerings:
 by email. by telephone.
 You can withdraw your marketing consent at any time by sending an email to netusage@us.ibm.com. Also you may unsubscribe from receiving marketing emails by clicking the unsubscribe link in each such email.
 More information on our processing can be found in the [IBM Privacy Statement](#). By submitting this form, I acknowledge that I have read and understand the IBM Privacy Statement.

I accept the product [Terms and Conditions](#) of this registration form.

Create Account

Privacy & Terms

Figure 56: Watson Signup

Once you have submitted the signup form an confirmation email will be sent to your email account, check your inbox and click on the confirm account link in the email you receive. This will activate your IBM Watson account. Once you have accepted the terms and conditions you will be taken to the product and service catalog of IBM Watson as shown in the image [Figure 57](#).

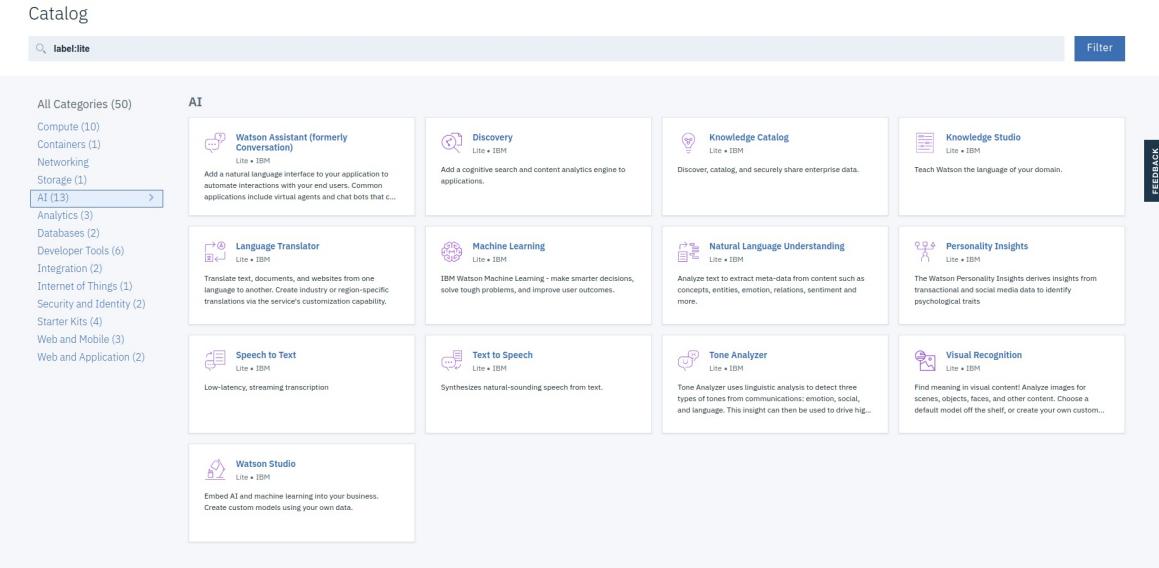


Figure 57: Watson Catalog

11.4.3 Understanding the free tier

IBM Watson provides a set of services for free with their Lite account. Since you did not provide any credit/debit card information when creating the account, by default you will have a Lite account. The lite plan does apply usage caps for services offered under the plan. If you need to expand and remove such limits you would have to upgrade to a payed account However the free quotas are typically more than sufficient for testing and learning purposes. For example under the Lite plan you can use the "Watson Assistant" service with caps such as 10K API calls per month.

11.5 GOOGLE IaaS CLOUD SERVICES

Google Cloud, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube. Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning. Registration requires a credit card or bank account details.

Google Cloud Platform provides Infrastructure as a Service, Platform as a Service, and Server-less Computing environments.

Google Cloud Platform is a part of Google Cloud, which includes the Google Cloud Platform public cloud infrastructure, as well as G Suite, enterprise versions of Android and Chrome OS, and application programming interfaces (APIs) for machine learning and Google Maps. See [Figure 58](#)

11.5.1 Cloud Computing Services and Products

Google Cloud Platform offers a full spectrum of cloud products and services for compute, storage, networking, big data, machine learning, operations, and more.



Compute

- [Choosing a Computing Option](#)
- [App Engine](#)
- [Compute Engine](#)
- [More](#)



Storage

- [Choosing a Storage Option](#)
- [Cloud Storage](#)
- [Cloud Spanner](#)
- [More](#)



Networking

- [Cloud DNS](#)
- [Cloud Interconnect](#)
- [CDN Interconnect](#)
- [Cloud Load Balancing](#)
- [More](#)



Big Data

- [BigQuery](#)
- [Cloud Composer](#)
- [Cloud Dataflow](#)
- [Cloud Dataproc](#)
- [More](#)



Machine Learning

- [Cloud TPU](#)
- [Cloud Machine Learning Engine](#)
- [Cloud Vision API](#)
- [Cloud Speech-to-Text API](#)
- [More](#)



Internet of Things

- [Cloud IoT Core](#)



Identity and Security

- [Auth Guide](#)
- [Cloud Identity and Access Management](#)
- [Cloud Identity-Aware Proxy](#)
- [Cloud Key Management Service](#)
- [Binary Authorization](#)
- [Cloud Data Loss Prevention API](#)
- [Cloud Armor](#)
- [More](#)



Operations

- [Stackdriver Logging](#)
- [Stackdriver Monitoring](#)
- [Stackdriver Error Reporting](#)
- [More](#)



Developer Tools

- [Cloud SDK](#)
- [Container Registry](#)
- [Cloud Build](#)
- [Cloud Source Repositories](#)
- [More](#)



Data Transfer

- [Transfer Appliance](#)
- [Storage Transfer Service](#)
- [Google BigQuery Data Transfer Service](#)



Resource Manager

- [Cloud Deployment Manager](#)
- [Cloud Marketplace](#)
- [More](#)

Figure 58: Google cloud services

11.6 OPENSTACK



11.6.1 Introduction

OpenStack can be described as a cloud operating system. OpenStack can be used on private or public clouds to manage large amounts of compute, storage and network resources. OpenStack is built up using a large number of small software components, which will be described in more detail in the next couple of sections. Another important aspect of OpenStack is that it is completely OpenSource, which means that anyone can access and use the product without having to pay any licensing or any other fee. And since the source code is publicly available under the Apache-2.0 License developers can modify and use OpenStack as needed.

OpenStack is managed and maintained by the “The OpenStack Foundation”, which is a non-profit organization which organizes the development of OpenStack and keeps the OpenStack community running.

11.6.2 OpenStack Architecture

OpenStack consists of a large number of small components. Which components to use for your OpenStack deployment depends on your usecases and requirements. The existing components can be integrated to achieve the desired deployment by carefully examining and understanding each component. [Figure 59](#) shows an high-level architecture diagram of OpenStack which gives a clear understanding of how the overall framework is organized.

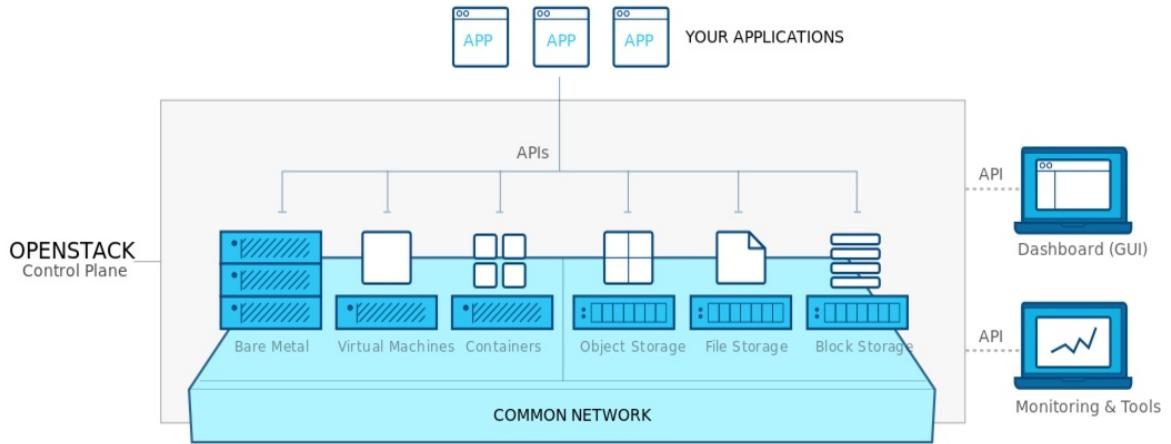


Figure 59: OpenStack Overview

Image reference - <https://www.openstack.org/software/>

However each of the high-level components are constructed using several small components. [Figure 60](#) shows one such popular deployment and its architecture.

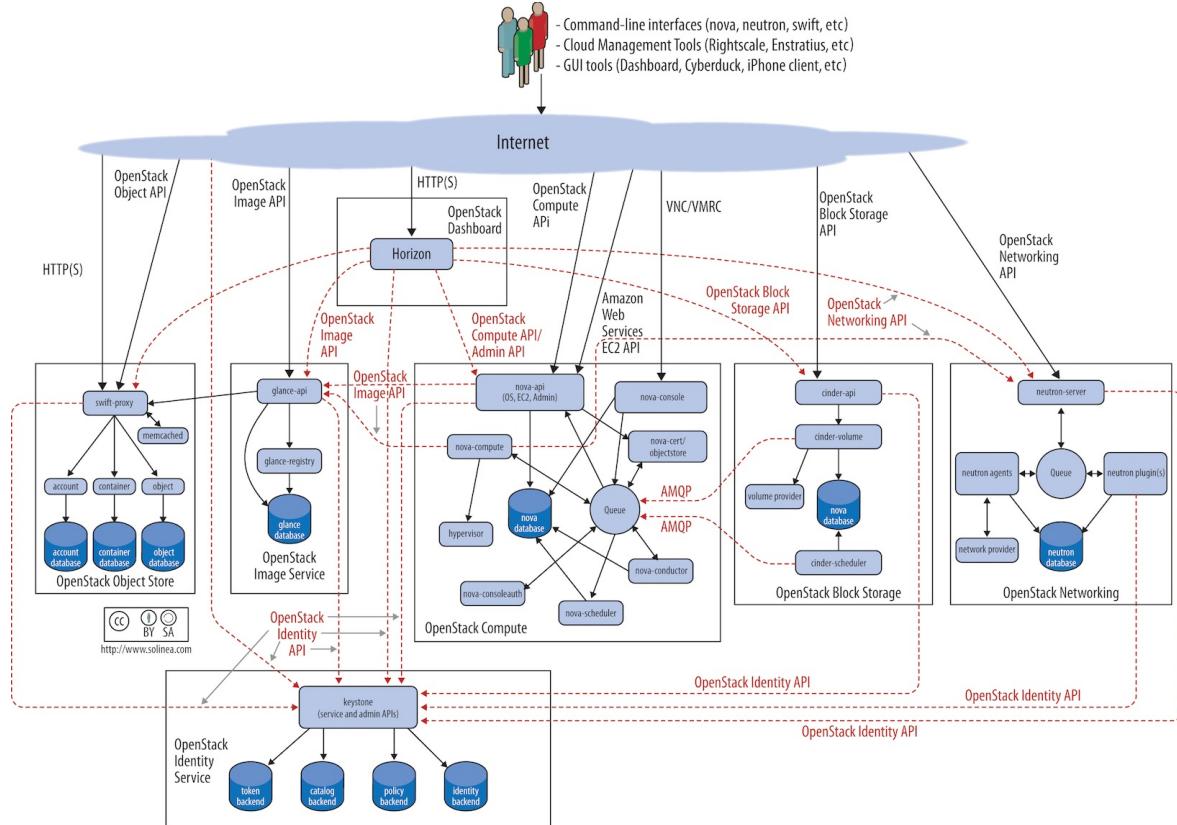


Figure 60: OpenStack Architecture

Image reference
<https://docs.openstack.org/arch-design/design.html>

- <https://docs.openstack.org/arch-design/design.html>

11.6.3 Components

The components in OpenStack can be divided into several sub-groups. And each group has several components which specialize in various tasks. The following list shows all the major component groups and components that are listed under each group. The list is referenced from the OpenStack documentation - [OpenStack Service List](https://docs.openstack.org/arch-design/design.html)

Compute

- NOVA - Compute Service
- ZUN - Containers Service
- QINLING - Functions Service

Bare Metal

- IRONIC - Bare Metal Provisioning Service
- CYBORG - Accelerators resource management

Storage

- SWIFT - Object store
- CINDER - Block Storage
- MANILA - Shared filesystems

Networking

- NEUTRON - Networking
- OCTAVIA - Load balancer
- DESIGNATE - DNS service

Shared Services

- KEYSTONE - Identity service
- GLANCE - Image service
- BARBICAN - Key management
- KARBOR - Application Data Protection as a Service
- SEARCHLIGHT - Indexing and Search

Orchestration

- HEAT - Orchestration
- SENLIN - Clustering service
- MISTRAL - Workflow service
- ZAQAR - Messaging Service
- BLAZAR - Resource reservation service
- AODH - Alarming Service

Workload Provisioning

- MAGNUM - Container Orchestration Engine Provisioning
- SAHARA - Big Data Processing Framework Provisioning
- TROVE - Database as a Service

Application Lifecycle

- MASAKARI - Instances High Availability Service
- MURANO - Application Catalog
- SOLUM - Software Development Lifecycle Automation
- FREEZER - Backup, Restore, and Disaster Recovery

API Proxies

- EC2API - EC2 API proxy

Web Frontend

- HORIZON - Dashboard

This list just includes the open stack services. Additionally, there are several other major component groups such as Operational Services, Add-Ons to Services and Bridges for Adjacent Tech listed in the services page at - [OpenStack Services](#).

11.6.4 Core Services

Among all the service components that are available for OpenStack, there are 9 services that are considered to be Core services, these services are essential to any OpenStack deployment.

11.6.4.1 Nova - Compute

Nova provides services to manage virtual machines in cloud environments. It is also capable of handling other compute resources such as containers and is highly scalable.

11.6.4.2 Glance - Image Services

Glance adds image services capabilities to OpenStack, this allows open stack users to manage virtual machine images and provides services such as image registration and discovery.

11.6.4.3 Swift - Object Storage

Swift allows developers refer to files and other data similar to object references. All actual storage and management is handled by Swift so the developers do not need to worry about where to store data and files.

11.6.4.4 Cinder - Block Storage

Cinder provides block storage management capabilities to OpenStack, Cinder supports several block storage devices underneath and provides an unified API so that developers do not need to worry or think about what device is been used underneath.

11.6.4.5 Neutron - Networking

Neutron provides networking capabilities to OpenStack. It allows the creation and management of various networks that are used as the communication medium for OpenStack deployments. Neutron supports multi-tenancy and scale to large deployments with ease. Extension frameworks for Neutron allow users to deploy more advance network features such as VPN's, firewalls, load-balancer, etc.

11.6.4.6 Horizon - Dashboard

Horizon is the graphical user interface(GUI) for OpenStack, which developers can use to manage and monitor their OpenStack deployment.

11.6.4.7 Keystone - Identity Service

Keystone is the identity management services in OpenStack, it keeps a list of users and maps all the access rights for each user for all the cloud services that are available in the OpenStack deployment. Keystone supports several authentication mechanisms such as classical user name password based authentication and token based

systems

11.6.4.8 Ceilometer - Telemetry

Ceilometer provides developers with billing and usage services that allow developers to bill end users based on each individual's usage amounts. It also records and saves usage values of the cloud for each user so that anything that needs verification can be done.

11.6.4.9 Heat - Orchestration

Heat is the orchestration component of OpenStack. It allows developers to use requirement files that define the resources requirements for cloud application, which can later be referenced when needed.

11.6.5 Access from Python and Scripts

11.6.5.1 Libcloud

Libcloud provides for some selected functionality a reasonable interface to OpenStack. More information is provided in Section [Python libcloud](#). More advanced resources are exposed through REST interfaces that may not be available in Libcloud. To access them new client libraries that are not included in libcloud need to be developed. Such functionality was exposed for example in FutureGrid to access cloud metric data.

11.6.5.2 DevStack

It is very convenient to be able to set up an OpenStack deployment for development purposes on our own single computer.

DevStack is a set of scripts that can be used by developers to manage and maintain their OpenStack development. DevStack was developed to increase the ease of use for developers. It is very useful to setup a

developer environment where you can test your deployment. More detailed information regarding DevStack can be found in their official documentaion - [DevStack documentation](#)

11.7 PYTHON LIBCLOUD



With all the cloud providers and cloud services that are currently available, it becomes hard to manage and maintain services that work with several services. Therefore it is good to have a unified service that allows developers to access many of the cloud services through a single abstraction. Apache Libcloud is a python library that was developed for this purpose. Apache Libcloud provides a unified API for many of the popular cloud providers and services.

Apache Libcloud currently supports many providers, the complete list of providers that are supported can be found at [Supported Providers](#)

However, it is good to keep in mind that the Libcloud API might not support some of the advanced features that are provided by some cloud services or some of the most recent features that are yet to be integrated into Libcloud

11.7.1 Service categories

Libcloud provides many services and defines several categories to distinguish between the main types of services. The list of categories is as bellow, More details about this list can be found at [Categories](#)
The list is extracted from the LibCloud documentation [LibCloud Docs](#)

- Cloud Servers and Block Storage - services such as Amazon EC2 and Rackspace CloudServers
- Cloud Object Storage and CDN - services such as Amazon S3 and Rackspace CloudFiles
- Load Balancers as a Service - services such as Amazon Elastic Load Balancer and GoGrid LoadBalancers
- DNS as a Service - services such as Amazon Route 53 and Zerigo

- Container Services - container virtualization like Docker and Rkt as well as container based services
- Backup as a Service - services such as Amazon EBS and OpenStack Freezer

each category has a set of terms that represent various constructs and services. For example the following list is the list of terms used in for Compute related services, this list is extracted from the [Compute docs](#)

11.7.1.0.1 Compute

- Node - represents a cloud or virtual server.
- NodeSize - represents node hardware configuration. Usually this is amount of the available RAM, bandwidth, CPU speed and disk size. Most of the drivers also expose an hourly price (in dollars) for the Node of this size.
- NodeImage - represents an operating system image.
- NodeLocation - represents a physical location where a server can be.
- NodeState - represents a node state. Standard states are: running, rebooting, terminated, pending, stopped, suspended, paused, erro, unknown.

11.7.1.0.2 Key Pair Management

- KeyPair - represents an SSH key pair object.

11.7.1.0.3 Block Storage

- StorageVolume - represents a block storage volume
- VolumeSnapshot - represents a point in time snapshot of a StorageVolume

You can find more complete information on Libcloud in the official documentations, this article will only provide a brief summary in most part: [Apache Libcloud Documentaions](#)

11.7.2 Installation

Libcloud can be installed via pip. Execute the following command in order to install Libcloud

```
pip install apache-libcloud
```

11.7.3 Quick Example

The following basic example shows you how the Python Libcloud library can be used to access information in a cloud provider

```
from pprint import pprint
import libcloud
cls = libcloud.get_driver(
    libcloud.DriverType.COMPUTE,
    libcloud.DriverType.COMPUTE.OPENSTACK)
driver = cls('username', 'api key')
pprint(driver.list_sizes())
pprint(driver.list_nodes())
```

11.7.4 Working with cloud services

In the following section we will look into how Libcloud can be used to perform various functions in specific cloud providers. One of the main aspects that change between different cloud providers is how authentication is done. Because of the unified API most of the other features are executed in the same manner.

11.7.4.1 Authenticating with cloud providers

Depending on the cloud provider, how Libcloud is granted access to your cloud account may differ, next we will look at some such examples

There are two main steps that are common to all providers

1. Using the `get_driver()` method to obtain a reference to the cloud provider driver

2. Instantiating the driver with the credentials to access the cloud

After you obtain the connection, it can be used to invoke various services

11.7.4.1.1 Amazon AWS

Authentication is performed for AWS as follows

```
from libcloud.compute.types import Provider
from libcloud.compute.providers import get_driver

EC2_ACCESS_ID = 'your access id'
EC2_SECRET_KEY = 'your secret key'

EC2Driver = get_driver(Provider.EC2)
conn = EC2Driver(EC2_ACCESS_ID, EC2_SECRET_KEY)
```

11.7.4.1.2 Azure

Authentication is performed for Azure as follows

```
from libcloud.compute.types import Provider
from libcloud.compute.providers import get_driver

# Azure related variables

AZURE_SUBSCRIPTION_ID = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
AZURE_MANAGEMENT_CERT_PATH = 'C:/Demo/azure_cert.pem'

AZDriver = get_driver(Provider.AZURE)
conn = AZDriver(subscription_id=AZURE_SUBSCRIPTION_ID, key_file=AZURE_MANAGEMENT_CERT_PATH)
```

11.7.4.1.3 OpenStack

Authentication is performed for OpenStack as follows

```
from libcloud.compute.providers import get_driver
from libcloud.compute.types import Provider

OpenstackDriver = get_driver(Provider.OPENSTACK)

#OpenStack related variables

OPENSTACK__AUTH_USERNAME = 'your_user_name'
OPENSTACK_AUTH_PASSWORD = 'your_auth_password'
```

```
conn = OpenStack(OPENSTACK__AUTH_USERNAME, OPENSTACK__AUTH_PASSWORD',
                 ex_force_auth_url='http://192.168.1.101:5000',
                 ex_force_auth_version='2.0_password')
```

11.7.4.2 Invoking services

In this section we will look into how we can use the connection created as instructed above to perform various services such as creating nodes, listing nodes, starting nodes and stopping nodes.

Appropriate authentication code as described in the previous section is assumed. This will give us an variable named conn which we will use for invoking Services. It is in the next sections not explicitly listed. It is indicated by our ... at the beginning

11.7.4.2.1 Creating Nodes

In this section we will look at the code that can be used to create a node in the provider a node which represents a virtual server

```
...
# retrieve available images and sizes
images = conn.list_images()

sizes = conn.list_sizes()

# create node with first image and first size
node = conn.create_node(name='yourservername', image=images[0], size=sizes[0])
```

11.7.4.2.2 Listing Nodes

In this section we will look at the code that can be used to list the nodes that have been created in the provider

```
...
nodes = conn.list_nodes()
print nodes
```

11.7.4.2.3 Starting Nodes

After the node (Virtual server) has been created the following code can be used to start the node

```
...
nodes = conn.list_nodes()
node = [n for n in nodes if 'yourservername' in n.name][0]
conn.ex_start(node=node)
```

11.7.4.2.4 Stoping Nodes

When needed the following command can be used to stop a node that has been started

```
...
nodes = conn.list_nodes()
node = [n for n in nodes if 'yourservername' in n.name][0]
conn.ex_stop(node=node)
```

11.7.5 Cloudmesh Community Program to Manage Clouds

As you have noticed since the authentication can change from cloud services to cloud service it would be much easier to use a simple python script to automatically handle the differences in the code.

We have provided such a python script which you can leverage to manage different cloud providers. You can find the python script and the corresponding .yaml file in the cloudmesh-community github repository.

- Python Script - <https://github.com/cloudmesh-community/cm/blob/master/cm.py>
- Yaml File - <https://github.com/cloudmesh-community/cm/blob/master/cloudmesh.yaml>

When using the script and yaml file please keep in mind the following steps to make sure you do not share your private keys and passwords on your publicly accessible Github account.

1. Create a folder in your computer that is not within a git clone that you have made. For example maybe you can use a new directory on your desktop
2. Copy the cm.py and cloudmesh.yaml files into this folder. Just to

make sure you are not working with the files under the git repo you should delete the cloudmesh.yaml file in that is in your local git repo.

3. change the needed fields in the yaml file and use the python script to access the cloud services using libcloud.

To illustrate how simple the program is and that it significantly improves your management of credentials we provide the following code:

```
from cm import cloudmesh

cm = cloudmesh()
cm.config()
driver = cm.get_driver("aws")
print("driver=", driver)
```

To switch to a different cloud, you just have to create it in the yaml file and use that name.

It will be your task to add more providers to it.

We intent to host the code sometime soon on pypi so you can issue the command

```
$ pip install cm-community
```

and this library will be installed for you.

11.7.6 Amazon Simple Storage Service S3 via libcloud

Next we explain how to use Amazon Web Services (AWS) S3 via libcloud. Apache libcloud is a python library that provides abstraction layer and hides the complexities of directly integrating with AWS API's, for that matter it allows you to do so for different cloud providers. In the sections below more detailed steps are shown to install and use libcloud for AWS S3.

11.7.6.1 Access key

To be able to access AWS S3 from libcloud we need the access key to be specified in the call. Access key can be setup on AWS console by navigating to `My Security credentials->Encryption Keys->Access Keys`.

11.7.6.2 Create a new bucket on AWS S3

In S3 you first need to create a bucket which is nothing but a container where you store your data in the form of files. This is where you can also define access controls.

- Click on S3 link on the AWS console under storage section, this will bring you to the create bucket window.
- Click on “Create Bucket” button, this opens up a wizard.
- Answer all mandatory questions on each page.
- Important point here is to note the “Bucket Name” and the “Region” you are creating this bucket in, as this information will be used while calling the API.

11.7.6.3 List Containers

List Containers function list all the containers of buckets available for the user in that particular region.

```
from libcloud.storage.types import Provider
from libcloud.storage.providers import get_driver

cls = get_driver(Provider.S3_US_EAST2)
driver = cls('api key', 'api secret key')

d = driver.list_containers();

print d;
```

11.7.6.4 List container objects

List container objects function shows the list of all objects in that container. Please note the output could be large depending on the files present in the bucket.

```

from libcloud.storage.types import Provider
from libcloud.storage.providers import get_driver

# Note I have used S3_US_EAST2 as this is the
# "region" where my S3 bucket is located.

cls = get_driver(Provider.S3_US_EAST2)
driver = cls('api key', 'api secret key')

container = driver.get_container(container_name='<bucket name>')

d = driver.list_container_objects(container);

print d;

```

11.7.6.5 Upload a file

Upload a file helps in uploading a local file to S3 bucket.

```

from libcloud.storage.types import Provider
from libcloud.storage.providers import get_driver

FILE_PATH = '/<file path>/<filename>'

# Note I have used S3_US_EAST2 as this is
# the "region" where my S3 bucket is located.

cls = get_driver(Provider.S3_US_EAST2)
driver = cls('api key', 'api secret key')

container = driver.get_container(container_name='<bucket name>')

extra = {'meta_data': {
    'owner': '<owner name>',
    'created': '2018-03-24'}}

with open(FILE_PATH, 'rb') as iterator:
    obj = driver.upload_object_via_stream(
        iterator=iterator,
        container=container,
        object_name='backup.tar.gz',
        extra=extra)

```

11.7.6.6 References

- <https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduct>
- Documentation about libcloud can be found at <https://libcloud.readthedocs.org>

- storage driver
http://libcloud.readthedocs.io/en/latest/_modules/libcl
- Examples:
<https://libcloud.readthedocs.io/en/latest/storage/exam>
- API
docs<http://libcloud.apache.org/apidocs/0.6.1/libcloud.s>

11.8 AWS Boto



Boto is a software development kit (SDK) that provides AWS interface for Python applications. It enables to write applications in Python that make use of Amazon Web Services.

Boto supports different AWS services such as, Elastic Compute Cloud (EC2), DynamoDB, AWS Config, CloudWatch and Simple Storage Service (S3).

In contrast to libcloud it only focusses to support AWS.

11.8.1 Boto versions

The current version of Boto is Boto3 and is available from:

- <https://github.com/boto/boto3>

The documentation from amazon is provided here:

- <http://aws.amazon.com/sdk-for-python>

It supports Python versions 2.6.5, 2.7 and 3.3+.

11.8.2 Boto Installation

To install boto with its latest release, use

```
$ pip install boto3
```

To install boto from source, use

```
$ git clone https://github.com/boto/boto3.git  
$ cd boto3
```

Before you install it we suggest that you either use pyenv or venv.

```
$ python setup.py install
```

To install additional modules to use boto.cloudsearch, boto.manage, boto.mashups and to get all modules required for test suite, than the run command

```
$ python setup.py install
```

11.8.3 Access key

An initial setup is required to be able to access AWS EC2 from BOTO wherein you provide the key and region details. You can find the key details from IAM console on AWS.

11.8.4 BOTO configuration

BOTO can be configured in two ways, either by using the aws configure command if you have AWS Command line interface installed or simply by manually creating and editing the `~/.aws/credentials` file to include below parameters.

```
[default]  
aws_access_key_id = <YOUR_ACCESS_KEY>  
aws_secret_access_key = <YOUR_SECRET_KEY>
```

Similar to libcloud, BOTO also requires the region where you would create your EC2 instance, the same can be maintained by creating a config file.

```
$ emacs .aws/config  
[default]  
region=<region name> # for example us-east
```

11.8.5 EC2 interface of Boto

11.8.5.0.1 Create connection

To access EC2 instance, first import the required package.

```
import boto3.ec2
```

Make a connection to from application by specifying AWS region in which the user account is created, aws access key and secret key. AWS provides access key and secret key when a new user is created. Access key and secret key helps to identify the user.

```
connection = boto3.ec2.connect_to_region('<region name>', aws_access_key_id =  
'<access key>', aws_secret_access_key = '<secret key>')
```

connection object now points to EC2Connection object returned by the function `connect_to_region`.

11.8.6 List EC2 instances

The code to list the running instances (if you have some) is very simple:

```
import boto3  
  
ec2 = boto3.client('ec2')  
response = ec2.describe_instances()  
print(response)
```

11.8.6.0.1 Launch a new instance

To launch a new instance with default properties

```
connection.run_instances('<ami-id>')
```

Additional parameters can be specified to create instance of specific type and security group.

```
connection.run_instances('<ami-id>', key_name='<key>', instance_type='<type>',  
security_groups=['<security group list>'])
```

Instance type specifies the storage and type of platform. Security groups are required to provide access rights such as access to SSH into the instance.

11.8.6.0.2 Check running instances

The `get_all_reservations` function of EC2Connection object will return list of running instances.

```
reservations = connection.get_all_reservations()  
instances = reservations[0].instances
```

11.8.6.0.3 Stop instance

Up and running instances can be stopped. Thw `stop_instances` function of connection object enables multiple instances to be stopped in one command.

```
connection.stop_instances(instance_ids=['<id1>', '<id2>', ...])
```

11.8.6.0.4 Terminate instance

To terminate one or more instances simultaneously, use the `terminate_instances` function.

```
connection.terminate_instances(instance_ids=['<id1>', '<id2>', ...])
```

11.8.6.1 Reboot instances

The next example showcases how to reboot an instance, which is copied from <http://boto3.readthedocs.io/en/latest/guide/ec2-example-managing-instances.html>

```
# Code copied from  
# http://boto3.readthedocs.io/en/latest/guide/ec2-example-managing-instances.html  
import boto3  
from botocore.exceptions import ClientError  
  
ec2 = boto3.client('ec2')  
  
try:  
    ec2.reboot_instances(InstanceIds=['INSTANCE_ID'], DryRun=True)  
except ClientError as e:  
    if 'DryRunOperation' not in str(e):  
        print("You don't have permission to reboot instances.")  
        raise  
try:  
    response = ec2.reboot_instances(InstanceIds=['INSTANCE_ID'], DryRun=False)  
    print('Success', response)  
except ClientError as e:  
    print('Error', e)
```

11.8.7 Amazon S3 interface of Boto

11.8.7.0.1 Create connection

Import required packages

```
import boto3.s3
from boto3.s3.key import Key
```

Create a connection

```
connection = boto.connect_s3('<access-key>', '<secret-key>')
```

11.8.7.0.2 Create new bucket in S3

Amazon S3 stores all its data in Bucket. There is no limitation specified by AWS about number of data files allowed per bucket.

Bucket name has to be unique name accross all the AWS regions and hence globally unique.

```
bucket = conn.create_bucket('<bucket_name>')
```

If bucket name is unique, a new bucket of specified name will get created. If bucket name is not unique, application will throw error as

```
boto.exception.S3CreateError: S3Error[409]: Conflict
```

11.8.7.0.3 Upload data

To upload a file in the S3 bucket, first create a key object from `new_key()` function of bucket.

```
key = bucket.new_key('hello2.txt')
key.set_contents_from_string('Hello World!')
```

This will create `hello.txt` file with content Hello World! in the text file. This file can be found inside the bucket in which new key is created.

11.8.7.0.4 List all buckets

One account can have maximum 100 buckets in which data objects can be stored.

```
result = connection.get_all_buckets()
```

The `get_all_buckets` function of `S3Connection` lists all the buckets within account. It returns `ResultSet` object which has list of all buckets.

11.8.7.0.5 List all objects in a bucket

Data objects stored in a bucket has a metadata associated with it such as `LastModified` date and time. This information can also be captured.

```
# To list files in selected bucket
for key in bucket.list():
    print "{name}".format(name = key.name)
    print "{size}".format(size = key.size)
    print "{modified}".format(modified = key.last_modified)
```

11.8.7.0.6 Delete object

To delete any data object from bucket, `delete_key` function of `bucket` is used.

```
k = Key(<bucket-name>, <file-name>)
k.delete()
```

11.8.7.0.7 Delete bucket

To delete a bucket, provide a bucket name and call the `delete_bucket` function of `S3Connection` object.

```
connection.delete_bucket('<bucket-name>')
```

11.8.8 References

- <https://github.com/boto/boto3>
- <https://boto3.readthedocs.io/en/latest/guide/quickstart.html#>
- <http://boto3.readthedocs.io/en/latest/guide/ec2-example-managing-instances.html>

11.8.9 Exercises

E.boto.cloudmesh.1:

will will nw create a cloudmesh tool that manages virtual machines on the commandline. For that we copy the code published at

- <https://boto3.amazonaws.com/v1/documentation/api/latest/example-managing-instances.html>.

Modify this code using docopts and look at samples in

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

where we use libcloud. The code from Amazon is.

```
import sys
import boto3
from botocore.exceptions import ClientError

instance_id = sys.argv[2]
action = sys.argv[1].upper()

ec2 = boto3.client('ec2')

if action == 'ON':
    # Do a dryrun first to verify permissions
    try:
        ec2.start_instances(InstanceIds=[instance_id], DryRun=True)
    except ClientError as e:
        if 'DryRunOperation' not in str(e):
            raise
    # Dry run succeeded, run start_instances without dryrun
    try:
        response = ec2.start_instances(InstanceIds=[instance_id], DryRun=False)
        print(response)
    except ClientError as e:
        print(e)
else:
    # Do a dryrun first to verify permissions
    try:
        ec2.stop_instances(InstanceIds=[instance_id], DryRun=True)
    except ClientError as e:
        if 'DryRunOperation' not in str(e):
            raise
    # Dry run succeeded, call stop_instances without dryrun
    try:
        response = ec2.stop_instances(InstanceIds=[instance_id], DryRun=False)
        print(response)
```

```
except ClientError as e:  
    print(e)
```

E.boto.cloudmesh.2:

Integrate, start, stop, reboot, and other useful functions

E.boto.cloudmesh.3:

Discuss the advantages of docopts.



12.1 INTRODUCTION TO MAPREDUCE



In this section we discuss about the background of Mapreduce along with Hadoop and core components of Hadoop.

We start out our section with a review of the python lambda expression as well as the map function. Understanding these concepts is helpful for our overall understanding of map reduce.

So before you watch the video, we encourage you to learn Sections {#s-python-lambda} and {#s-python-map}.

Now that you have a basic understanding of the map function we recommend to watch our videos about mapreduce, hadoop and spark which we provide within this chapter.

[Map Reduce, Hadoop, and Spark \(19:02\) Hadoop A](#)

MapReduce is a programming technique or processing capability which operates in a cluster or a grid on a massive data set and brings out reliable output. It works on essentially two main functions – map() and reduce(). MapReduce processes large chunks of data so its highly beneficial to operate in multi-threaded fashion meaning parallel processing. MapReduce can also take advantage of data locality so that we do not loose much on communication of data from place to another.

12.1.1 MapReduce Algorithm

MapReduce can operate on a filesystem, which is an unstructured data or a database, a structured data and these are the following three stages of its operation (see [Figure 61](#)):

1. **Map:** This method processes the very initial data set.

Generally, the data is in file format which can be stored in HDFS (Hadoop File System). Map function reads the data line by line and creates several chunks of data and that is again stored in HDFS. This broken set of data is in key/value pairs. So in multi-threaded environment, there will be many worker nodes operating on the data using this map() function and write this intermediate data in form of key/value to temporary data storage.

2. **Shuffle:** In this stage, worker nodes will shuffle or redistribute the data in such a way that there is only one copy for each key.
3. **Reduce:** This function always comes at last and it works on the data produced by map and shuffle stages and produces even smaller chunk of data which is used to calculate output.

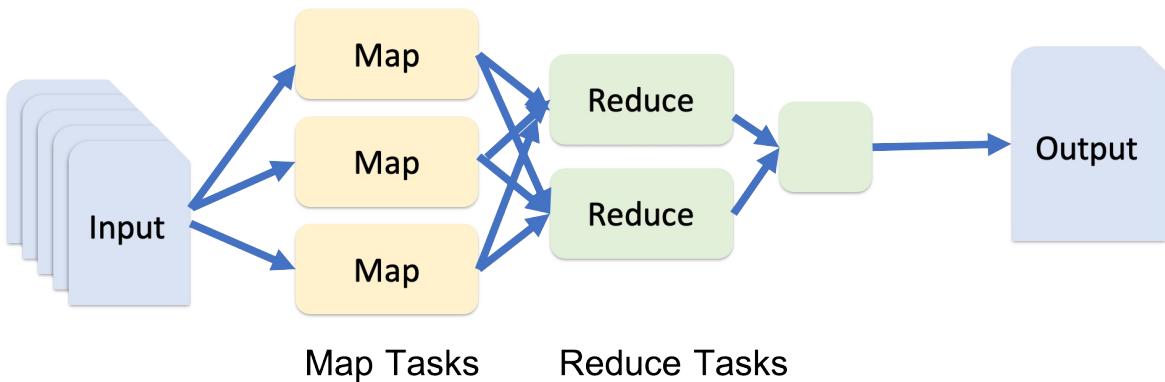


Figure 61: MapReduce Conceptual diagram

The Shuffle operation is very important here as that is mainly responsible for reducing the communication cost. The main advantage of using MapReduce algorithm is that it becomes very easy to scale up data processing just by adding some extra computing nodes. Building up map and reduce methods are sometimes nontrivial but once done, scaling up the applications is so easy that it is just a matter of changing configuration. Scalability is really big advantage of MapReduce model. In the traditional way of data processing, data was moved from nodes to the master and then the processing happens in master machine. In this approach, we lose bandwidth and time on moving data to master and parallel operation cannot happen. Also master can get over-burdened and fail. In

MapReduce approach, Master node distributes the data to the worker machines which are in themselves a processing unit. So all worker process the data in parallel and the time taken to process the data is reduced tremendously. (see [Figure 62](#))

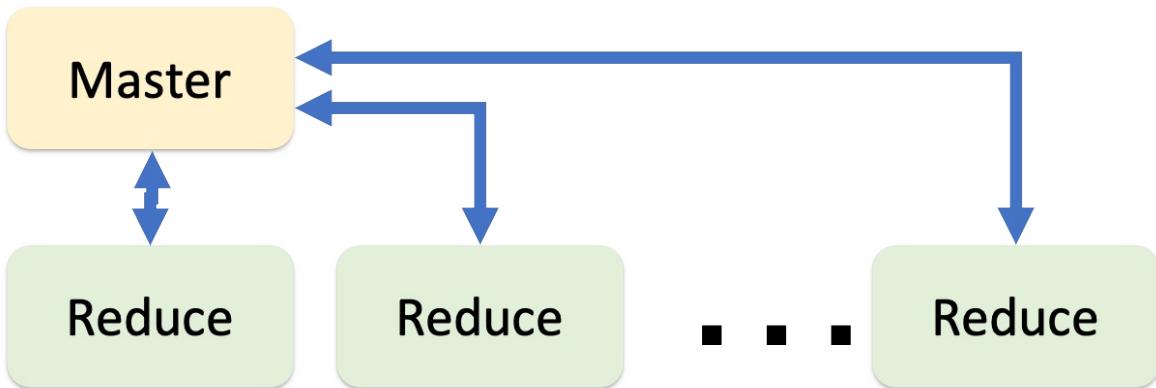


Figure 62: MapReduce Master worker diagram

12.1.1.1 MapReduce Example: Word Count

Let us understand MapReduce by an example. For example: we have a text file as Sample.txt as Cat, Bear, Camel, Bird, Cat, Bird, Camel, Cat, Bear, Camel, Cat, Camel

1. First we divide the input into four parts so that individual nodes can handle the load.
2. We tokenize each word and assign weightage of value “1” to each word.
3. This way we will have a list of key-value pairs with key being the word and value as 1.
4. After this mapping phase, shuffling phase starts where all maps with same key are sent corresponding reducer.
5. Now each reducer will have a unique key and a list of values for each key which in this case is all 1s.
6. After that, each reducer will count the total number of 1s and assigns final count to each word.
7. The final output is then written to a file. (see [Figure 63](#))

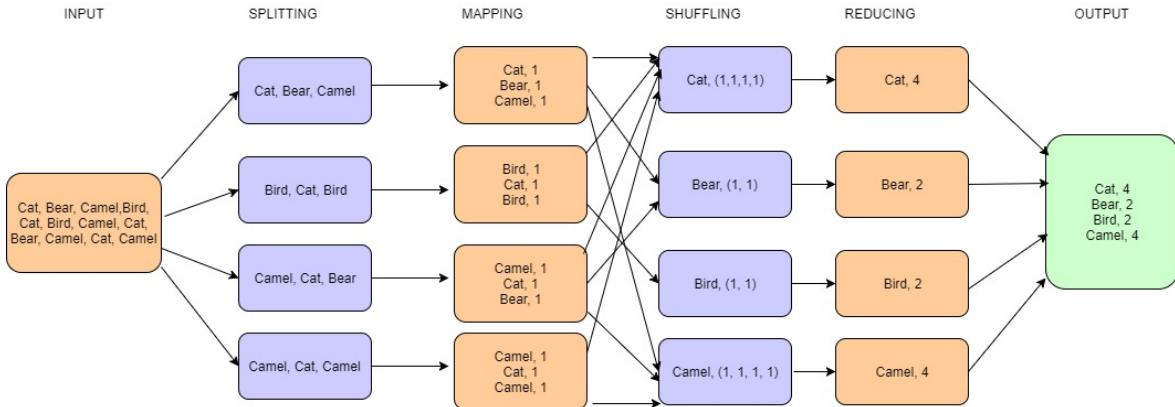


Figure 63: MapReduce WordCount [30]

Let us see an example of map() and reduce() methods in code for this word count example.

```
public static class Map extends Mapper<LongWritable,
                                         Text,
                                         Text,
                                         IntWritable> {

    public void map(LongWritable key,
                    Text value,
                    Context context)
        throws IOException, InterruptedException {

        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            value.set(tokenizer.nextToken());
            context.write(value, new IntWritable(1));
        }
    }
}
```

Here we have created a class **Map** which extends **Mapper** from MapReduce framework and we override **map()** method to declare the key/value pairs. Next, there will be a reduce method defined inside **Reduce** class as below and both input and output here is a key/value pairs:

```
public static class Reduce extends Reducer<Text,
                                         IntWritable,
                                         Text, IntWritable> {

    public void reduce(Text key,
                      Iterable<IntWritable> values,
                      Context context)
        throws IOException, InterruptedException {

        int sum=0;
```

```

        for(IntWritable x: values) {
            sum+=x.get();
        }
        context.write(key, new IntWritable(sum));
    }
}

```

12.1.2 Hadoop MapReduce and Hadoop Spark

In earlier version of Hadoop, we could use MapReduce with HDFS directly but from 2.0 onwards, YARN(Cluster Resource Management) is introduced which acts as a layer between MapReduce and HDFS and using this YARN, many other BigData frameworks can connect to HDFS as well. (see [Figure 64](#))

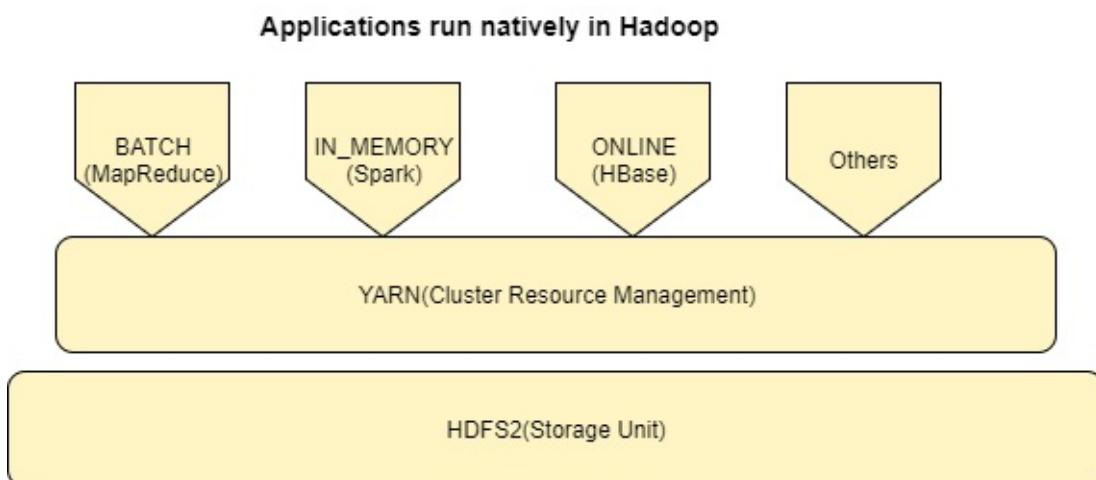


Figure 64: MapReduce Hadoop and Spark [\[31\]](#)

There are many big data frameworks available and there is always a question as to which one is the right one. Leading frameworks are Hadoop MapReduce and Apache Spark and choice depends on business needs. Let us start comparing both of these frameworks with respect to their processing capability.

12.1.2.1 Apache Spark

Apache Spark is lightning fast cluster computing framework. Spark is in-memory system. Spark is 100 time faster than Hadoop MapReduce.

12.1.2.2 Hadoop MapReduce

Hadoop MapReduce reads and writes on disk because of this it is a slow system and that affects the volume of data been processed. But Hadoop is a scalable and fault tolerant, it us good for linear processing.

12.1.2.3 Key Differences

The key differences between them are as follows:

1. **Speed:** Spark is lightning fast cluster computing framework and operates up to 100 time faster in-memory and 10 times faster than Hadoop on disk. In-memory processing reduces the disk read/write processes which are time consuming.
2. **Complexity:** Spark is easy to use since there are many APIs available but for Hadoop, developers need to code the functions which makes it harder.
3. **Application Management:** Spark can perform batch processing, interactive and Machine Learning and Streaming of data, all in the same cluster, which makes it a complete framework for data analysis whereas Hadoop is just a batch engine and it requires other frameworks for other tasks which makes it somewhat difficult to manage.
4. **Real-Time Data Analysis** Spark is capable of processing real time data with great efficiency. But Hadoop was designed primarily for batch processing so it cannot live data.
5. **Fault Tolerance:** Both the systems are fault tolerant so there is no need to restart the applications from scratch.
6. **Data Volume:** As the data for spark is held in memory larger data volumes are better managed in Hadoop.

12.1.3 References

- [32] <https://www.ibm.com/analytics/hadoop/mapreduce>
- [33] <https://en.wikipedia.org/wiki/MapReduce>
- [34]

- [30] https://www.edureka.co/blog/mapreduce-tutorial/?utm_source=youtube&utm_campaign=mapreduce-tutorial-161216-wr&utm_medium=description
- [35] <https://www.quora.com/What-is-the-difference-between-Hadoop-and-Spark>
- [36] <https://data-flair.training/blogs/apache-spark-vs-hadoop-mapreduce>
- [31] <https://www.youtube.com/watch?v=SqvAaB3vK8U&list=WL&index=25&t=2547s>

12.2 HADOOP



Hadoop is an open source framework for storage and processing of large datasets on commodity clusters. Hadoop internally uses its own file system called HDFS (Hadoop Distributed File System).

The motivation for Hadoop was introduced in Section Mapreduce

12.2.1 Hadoop and MapReduce

In this section we discuss about the usage Hadoop MapReduce architecture.

[Hadoop 13:19 Hadoop B](#)

12.2.2 Hadoop EcoSystem

In this section we discuss about the Hadoop EcoSystem and the architecture.

[Hadoop 12:57 Hadoop C](#)

12.2.3 Hadoop Components

In this section we discuss about Hadoop Components in detail.

 [Hadoop 15:14 Hadoop D](#)

12.2.4 Hadoop and the Yarn Resource Manager

In this section we discuss about Yarn resource manager and novel components added to the Hadoop framework in case of improving the performance and minimizing fault tolerance.

 [Hadoop 14:55 Hadoop E](#)

12.2.5 PageRank

In this section we discuss about a real world problem that can be solved using the MapReduce technique. PageRank is a problem solved by the earliest stages of the Google.inc. In this section we discuss about the theoretical background about this problem and we discuss how this can be solved using the map reduce concepts.

 [Hadoop 15:14 Hadoop E](#)

12.2.6 INSTALLATION OF HADOOP



○ THis section was checked for hadoop 3.1.1 in Ubuntu 18.04. We also describe the installation of the Yarn resource manager. We assume that you have ssh, and rsync installed and use emacs as editor.

12.2.6.1 Releases

Hadoop changes on regular basis. Before follwoing this section, we recommend that you visit

- <https://hadoop.apache.org/releases.html>

The list of downloadable files is also available at

and verify that you use an up to dat version.If the verison of this

instalation is outdated. we ask you as excrsise to update it.

12.2.6.2 Prerequisites

```
sudo apt-get install ssh  
sudo apt-get install rsync  
sudo apt-get install emacs
```

12.2.6.3 User and User Group Creation

For security reasons we will install hadoop in a particular user and user group. We will use the following

```
sudo addgroup hadoop_group  
sudo adduser --ingroup hadoop_group hduser  
sudo adduser hduser sudo
```

These steps will provide sudo privileges to the created hduser user and add the user to the group `hadoop_group`.

12.2.6.4 Configuring SSH

Here we configure SSH key for the local user to install hadoop with a ssh-key. This is different from the ssh-key you used for Github, FutureSystems, etc. Follow this section to configure it for Hadoop installation.

The ssh content is included here because, we are making a ssh key for this specific user. Next, we have to configure ssh to be used by the hadoop user.

```
sudo su - hduser  
ssh-keygen -t rsa
```

Follow the instructions as provided in the commandline. When you see the following console input, press ENTER. Here only we will create password less keys. IN general this is not a good idea, but for this case we make an exception.

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

Next you will be asked to enter a password for ssh configuration,

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

Here enter the same password

```
Enter same passphrase again:
```

Finally you will see something like this after these steps are finished.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0UBCPd6oYp7MEzCp0hMhNiJyQo6PaPCDu0T48xUDDc0 hduser@computer
The key's randomart image is:
+---[RSA 2048]---+
|   .+000
| . oE.oo
|+ ... .+.
|X+= . o..
|XX.o o.S
|Bo+ + .o
|*o * +.
|*... *.
| +.o..
+---[SHA256]---
```

You have successfully configured ssh.

12.2.6.5 Installation of Java

If you are already logged into su, you can skip the next command:

```
su - hduser
```

Now execute the following commands to download and install java

```
mkdir -p ~/cloudmesh/bin
cd ~/cloudmesh/bin
wget -c --header "Cookie: \
oraclelicense=accept-securebackup-cookie" \
"http://download.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jdk-8u191-linux-x64.tar.gz"
tar xvzf jdk-8u191-linux-x64.tar.gz
```

Please note that users must accept Oracle OTN license before downloading JDK.

12.2.6.6 Installation of Hadoop

First we will take a look on how to install Hadoop 3.1.1 on Ubuntu 16.04. We may need a prior folder structure to do the installation properly.

```
cd ~/cloudmesh/bin/  
wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-3.1.1/hadoop-3.1.1.tar.gz  
tar -xzvf hadoop-3.1.1.tar.gz
```

12.2.6.7 Hadoop Environment Variables

In Ubuntu the environmental variables are setup in a file called bashrc at it can be accessed the following way

```
emacs ~/.bashrc
```

Now add the following to your `~/.bashrc` file

```
export JAVA_HOME=~/cloudmesh/bin/jdk1.8.0_191  
export HADOOP_HOME=~/cloudmesh/bin/hadoop-3.1.1  
export YARN_HOME=$HADOOP_HOME  
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop  
export PATH=$HADOOP_HOME/bin:$JAVA_HOME/bin:$PATH
```

In Emacs to save the file `ctrl-x-s` and `ctrl-x-c` to exit. After editing you must update the variables in the system.

```
source ~/.bashrc  
java -version
```

If you have installed things properly there will be no errors. It will show the version as follows,

```
java version "1.8.0_191"  
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

And verifying the hadoop installation,

hadoop

If you have successfully installed this, there must be a message shown as below.

```
Usage: hadoop [--config confdir] COMMAND
  where COMMAND is one of:
    fs                  run a generic filesystem user client
    version             print the version
    jar <jar>            run a jar file
    checknative [-a|-h]  check native hadoop and compression libraries availability
    distcp <srcurl> <desturl> copy file or directories recursively
    archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
    classpath           prints the class path needed to get the
    credential          interact with credential providers
    daemonlog           Hadoop jar and the required libraries
    trace               get/set the log level for each daemon
    view and modify Hadoop tracing settings
    or
    CLASSNAME          run the class named CLASSNAME

  Most commands print help when invoked w/o parameters.
```

12.2.7 HADOOP VIRTUAL CLUSTER INSTALLATION USING CLOUDMESH

 This version is dependent on an older version of cloudmesh

 :TODO: we need to add teh instalation instructions based on this version

12.2.7.1 Cloudmesh Cluster Installation

Before you start this lesson, you MUST finish cm_install.

This lesson is created and test under the newest version of Cloudmesh client. Update yours if not.

To manage virtual cluster on cloud, the command is `cm cluster`. Try `cm cluster help` to see what other commands are and what options they supported.

12.2.7.1.1 Create Cluster

To create a virtual cluster on cloud, we must define an active cluster

specification with `cm cluster define` command. For example, we define a cluster with 3 nodes:

```
$ cm cluster define --count 3
```

All options will use the default setting if not specified during cluster

define. Try `cm cluster help` command to see what options `cm cluster define` has and means, here is part of the usage information::

```
$ cm cluster help

usage: cluster create [-n NAME] [-c COUNT] [-C CLOUD] [-u NAME] [-i IMAGE] [-f FLAVOR] [-k KEY] [-s NAME] [-AI]

Options:
-A --no-activate Do not activate this cluster
-I --no-floating-ip Do not assign floating IPs
-n NAME --name=NAME Name of the cluster
-c COUNT --count=COUNT Number of nodes in the cluster
-C NAME --cloud=NAME Name of the cloud
-u NAME --username=NAME Name of the image login user
-i NAME --image=NAME Name of the image
-f NAME --flavor=NAME Name of the flavor
-k NAME --key=NAME Name of the key
-s NAME --secgroup=NAME NAME of the security group
-o PATH --path=PATH Output to this path ...
```

Floating IP is a valuable and limited resource on cloud.

`cm cluster define` will assign floating IP to every node within the cluster by default. Cluster creation will fail if the floating IPs run out on cloud. When you run into error like this, use option `-I` or `--no-floating-ip` to avoid assigning floating IPs during cluster creation:

```
$ cm cluster define --count 3 --no-floating-ip
```

Then manually assign floating IP to one of the nodes. Use this node as a logging node or head node to log in to all the other nodes.

We can have multiple specifications defined at the same time. Every time a new cluster specification is defined, the counter of the default cluster name will increment. Hence, the default cluster name will be `cluster-001`, `cluster-002`, `cluster-003` and so on. Use `cm cluster avail` to check all

the available cluster specifications:

```
$ cm cluster avail
cluster-001
  count          : 3
  image          : CC-Ubuntu14.04
  key            : xl41
  flavor          : m1.small
  secgroup        : default
  assignFloatingIP : True
  cloud           : chameleon
> cluster-002
  count          : 3
  image          : CC-Ubuntu14.04
  key            : xl41
  flavor          : m1.small
  secgroup        : default
  assignFloatingIP : False
  cloud           : chameleon
```

With `cm cluster use [NAME]`, we are able to switch between different specifications with given cluster name:

```
$ cm cluster use cluster-001
$ cm cluster avail
> cluster-001
  count          : 3
  image          : CC-Ubuntu14.04
  key            : xl41
  flavor          : m1.small
  secgroup        : default
  assignFloatingIP : True
  cloud           : chameleon
cluster-002
  count          : 3
  image          : CC-Ubuntu14.04
  key            : xl41
  flavor          : m1.small
  secgroup        : default
  assignFloatingIP : False
  cloud           : chameleon
```

This will activate specification `cluster-001` which assigns floating IP during creation rather than the latest one `cluster-002`.

With our cluster specification ready, we create the cluster with command `cm cluster allocate`. This will create a virtual cluster on the cloud with the activated specification:

```
$ cm cluster allocate
```

Each specification can have one active cluster, which means `cm cluster`

`allocate` does nothing if there is a successfully active cluster.

12.2.7.1.2 Check Created Cluster

With command `cm cluster list`, we can see the cluster with the default name `cluster-001` we just created:

```
$ cm cluster list
cluster-001
```

Using `cm cluster nodes [NAME]`, we can also see the nodes of the cluster along with their assigned floating IPs of the cluster:

```
$ cm cluster nodes cluster-001
x141-001 129.114.33.147
x141-002 129.114.33.148
x141-003 129.114.33.149
```

If option `--no-floating-ip` is included during definition, you will see nodes without floating IP:

```
$ cm cluster nodes cluster-002
x141-004 None
x141-005 None
x141-006 None
```

To log in one of them, use command `cm vm assign IP [NAME]` to assign a floating IP to one of them:

```
$ cm vm ip assign x141-006
$ cm cluster nodes cluster-002
x141-004 None
x141-005 None
x141-006 129.114.33.150
```

Then you can log in this node as a head node of your cluster by `cm vm ssh [NAME]`:

```
$ cm vm ssh x141-006
cc@x141-006 $
```

12.2.7.1.3 Delete Cluster

Using `cm cluster delete [NAME]`, we are able to delete the cluster we created:

```
$ cm cluster delete cluster-001
```

Option `--all` can delete all the clusters created, so be careful:

:

```
$ cm cluster delete -all
```

Then we need to undefine our cluster specification with command `cm cluster undefine [NAME]`:

```
$ cm cluster undefine cluster-001
```

Option `--all` can delete all the cluster specifications:

```
$ cm cluster undefine --all
```

12.2.7.2 Hadoop Cluster Installation

This section is built upon the previous one. Please finish the previous one before start this one.

12.2.7.2.1 Create Hadoop Cluster

To create a Hadoop cluster, we need to first define a cluster with `cm cluster define` command:

```
$ cm cluster define --count 3
```

To deploy a Hadoop cluster, we only support image `cc-ubuntu14.04`

on Chameleon. DO NOT use `cc-ubuntu16.04` or any other images. You will need to specify it if it's not the default image:

```
$ cm cluster define -count 3 -image CC-Ubuntu14.04
```

Then we define the Hadoop cluster upon the cluster we defined using `cm hadoop define` Command:

```
$ cm hadoop define
```

Same as `cm cluster define`, you can define multiple specifications for the Hadoop cluster and check them with `cm hadoop avail`:

```
$ cm hadoop avail
> stack-001
  local_path          : /Users/tony/.cloudmesh/stacks/stack-001
  addons              : []
```

We can use `cm hadoop use [NAME]` to activate the specification with the given name:

```
$ cm hadoop use stack-001
```

May not be available for current version of Cloudmesh Client.

Before deploy, we need to use `cm hadoop sync` to checkout / synchronize the Big Data Stack from Github.com:

```
$ cm hadoop sync
```

To avoid errors, make sure you are able to connect to Github.com using SSH:

<https://help.github.com/articles/connecting-to-github-with-ssh/>.

Finally, we are ready to deploy our Hadoop cluster:

```
$ cm hadoop deploy
```

This process could take up to 10 minutes based on your network.

To check Hadoop is working or not. Use `cm vm ssh` to log into the `Namenode` of the Hadoop cluster. It's usually the first node of the cluster:

```
$ cm vm ssh node-001
cc@hostname$
```

Switch to user `hadoop` and check HDFS is set up or not:

```
cc@hostname$ sudo su - hadoop
hadoop@hostname$ hdfs dfs -ls /
Found 1 items
drwxrwx---  - hadoop hadoop, hadoopadmin          0 2017-02-15 17:26 /tmp
```

Now the Hadoop cluster is properly installed and configured.

12.2.7.2.2 Delete Hadoop Cluster

To delete the Hadoop cluster we created, use command `cm cluster delete [NAME]` to delete the cluster with given name:

```
$ cm cluster delete cluster-001
```

Then undefine the Hadoop specification and the cluster specification:

```
$ cm hadoop undefine stack-001  
$ cm cluster undefine cluster-001
```

May not be available for current version of Cloudmesh Client.

12.2.7.3 Advanced Topics with Hadoop

12.2.7.3.1 Hadoop Virtual Cluster with Spark and/or Pig

To install Spark and/or Pig with Hadoop cluster, we first use command `cm hadoop define` but with `ADDON` to define the cluster specification.

For example, we create a 3-node Spark cluster with Pig. To do that, all we need is to specify `spark` as an `ADDON` during Hadoop definition:

```
$ cm cluster define --count 3  
$ cm hadoop define spark pig
```

Using `cm hadoop addons`, we are able to check the current supported addon:

```
$ cm hadoop addons
```

With `cm hadoop avail`, we can see the detail of the specification for the Hadoop cluster:

```
$ cm hadoop avail  
> stack-001  
local_path : /Users/tony/.cloudmesh/stacks/stack-001  
addons : [u'spark', u'pig']
```

Then we use `cm hadoop sync` and `cm hadoop deploy` to deploy our Spark cluster:

```
$ cm hadoop sync  
$ cm hadoop deploy
```

This process will take 15 minutes or longer.

Before we proceed to the next step, there is one more thing we need to, which is to make sure we are able to ssh from every node to others without password. To achieve that, we need to execute `cm cluster cross_ssh`:

```
$ cm cluster cross_ssh
```

12.2.7.3.2 Word Count Example on Spark

Now with the cluster ready, let's run a simple Spark job, Word Count, on one of William Shakespeare's work. Use `cm vm ssh` to log into the `Namenode` of the Spark cluster. It should be the first node of the cluster:

```
$ cm vm ssh node-001  
cc@hostname$
```

Switch to user `hadoop` and check HDFS is set up or not:

```
cc@hostname$ sudo su - hadoop  
hadoop@hostname$
```

Download the input file from the Internet:

```
wget --no-check-certificate -O inputfile.txt \  
https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt
```

You can also use any other text file you preferred. Create a new directory `wordcount` within HDFS to store the input and output:

```
$ hdfs dfs -mkdir /wordcount
```

Store the input text file into the directory:

```
$ hdfs dfs -put inputfile.txt /wordcount/inputfile.txt
```

Save the following code as `wordcount.py` on the local file system on

Namenode:

```
import sys

from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

    # take two arguments, input and output
    if len(sys.argv) != 3:
        print("Usage: wordcount <input> <output>")
        exit(-1)

    # create Spark context with Spark configuration
    conf = SparkConf().setAppName("Spark Count")
    sc = SparkContext(conf=conf)

    # read in text file
    text_file = sc.textFile(sys.argv[1])

    # split each line into words
    # count the occurrence of each word
    # sort the output based on word
    counts = text_file.flatMap(lambda line: line.split(" ")) \
        .map(lambda word: (word, 1)) \
        .reduceByKey(lambda a, b: a + b) \
        .sortByKey()

    # save the result in the output text file
    counts.saveAsTextFile(sys.argv[2])
```

Next submit the job to Yarn and run in distribute:

```
$ spark-submit --master yarn --deploy-mode client --executor-memory 1g \
--name wordcount --conf "spark.app.id=wordcount" wordcount.py \
hdfs://192.168.0.236:8020/wordcount/inputfile.txt \
hdfs://192.168.0.236:8020/wordcount/output
```

Finally, take a look at the result in the output directory:

```
$ hdfs dfs -ls /wordcount/outputfile/
Found 3 items
-rw-r--r-- 1 hadoop hadoop, hadoopadmin          0 2017-03-07 21:28 /wordcount/output/_SUCCESS
-rw-r--r-- 1 hadoop hadoop, hadoopadmin 483182 2017-03-07 21:28 /wordcount/output/part-00000
-rw-r--r-- 1 hadoop hadoop, hadoopadmin 639649 2017-03-07 21:28 /wordcount/output/part-00001
$ hdfs dfs -cat /wordcount/output/part-00000 | less
(u'', 517065)
(u'', 241)
(u"\'Tis", 1)
(u"A", 4)
(u"AS-IS", 1)
(u"Air,", 1)
(u"Alas,", 1)
(u"Amen", 2)
(u"Amen?", 1)
```

```
(u'"Amen,"', 1)
```

...

12.3 SPARK



12.3.1 SPARK LECTURES



This section covers an introduction to Spark that is split up into eight parts. We discuss Spark background, RDD operations, Shark, Spark ML, Spark vs Other Frameworks.

12.3.1.1 Motivation for Spark

In this section we discuss about the background of Spark and core components of Spark.

[Spark 15:57 Spark A](#)

12.3.1.2 Spark RDD Operations

In this section we discuss about the background of RDD operations along with other transformation functionality in Spark.

[Spark 12:17 Spark B](#)

12.3.1.3 Spark DAG

In this section we discuss about the background of DAG (direct acyclic graphs) operations along with other components like Shark in the earlier stages of Spark.

[Spark 10:37 Spark C](#)

12.3.1.4 Spark vs. other Frameworks

In this section we discuss about the real world applications that can

be done using Spark. And also we discuss some comparision results obtained from experiments done in Spark along with Frameworks like Harp, Harp DAAL, etc. We discuss the benchmarks and performance obtained from such experiments.

[Spark 26:18 Spark D](#)

12.3.2 INSTALLATION OF SPARK



In this section we will discuss how to install Spark 2.3.2 in Ubuntu 18.04.

12.3.2.1 Prerequisites

We assume that you have ssh, and rsync installed and use emacs as editor.

```
sudo apt-get install ssh  
sudo apt-get install rsync  
sudo apt-get install emacs
```

12.3.2.2 Installation of Java

First download Java 8.

```
mkdir -p ~/cloudmesh/bin  
cd ~/cloudmesh/bin  
wget -c --header "Cookie: oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn/java/jdk/8u161-b12/b309b3362ce347f7bd5d57f5491f7fe/jdk-8u161-linux-x64.tar.gz"  

```

Then add the environmental variables to the bashrc file.

```
emacs ~/.bashrc  
  
export JAVA_HOME=~/cloudmesh/bin/jdk1.8.0_161  
export PATH=$JAVA_HOME/bin:$PATH
```

Source the bashrc file after adding the environmental variables.

```
source ~/.bashrc
```

12.3.2.3 Install Spark with Hadoop

Here we use Spark packaged with Hadoop. In this package Spark uses Hadoop 2.7.0 in the packaged version. Note that in Section [Hadoop Installation](#) we use for the vanilla Hadoop installation Hadoop 3.0.1.

Create the base directories and go to the directory.

```
mkdir -p ~/cloudmesh/bin  
cd ~/cloudmesh/bin
```

Then download Spark 2.3.2 as follows.

```
wget https://archive.apache.org/dist/spark/spark-2.3.2/spark-2.3.2-bin-hadoop2.7.tgz
```

Now extract the file,

```
tar xzf spark-2.3.2-bin-hadoop2.7.tgz  
mv spark-2.3.2-bin-hadoop2.7 spark-2.3.2
```

12.3.2.4 Spark Environment Variables

Open up bashrc file and add environmental variables as follows.

```
emacs ~/.bashrc
```

Go to the last line and add the following content.

```
export SPARK_HOME=~/cloudmesh/bin/spark-2.3.2  
export PATH=$SPARK_HOME/bin:$PATH
```

Source the bashrc file.

```
source ~/.bashrc
```

12.3.2.5 Test Spark Installation

Open up a new terminal and then run the following command.

```
spark-shell
```

If it has been configured properly, it will display the following content.

```
Spark context Web UI available at http://192.168.1.66:4041
Spark context available as 'sc' (master = local[*], app id = local-1521674331361).
Spark session available as 'spark'.
Welcome to

    / _ \
   / \ \_ \ \_ \ \_ \ \_ \
  / \ \_ \ \_ \ \_ \ \_ \
 / \ \_ \ \_ \ \_ \ \_ \
/ \ \_ \
version 2.3.2

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_151)
Type in expressions to have them evaluated.
Type :help for more information.
```

Please check the console LOGS and find the port number on which the Spark Web UI is hosted. It will show something like:

Spark context Web UI available at: <some url>

Then take a look the following address in the browser.

```
http://localhost:4041
```

If you see the Spark Dashboard, then you can realize you have installed Spark successfully.

12.3.2.6 Install Spark With Custom Hadoop

Installing Spark with pre-existing Hadoop version is favorable, if you want to use the latest features from the latest Hadoop version or when you need a specific Hadoop version depending on the external dependencies to your project.

First we need to download the Spark packaged without Hadoop.

```
mkdir -p ~/cloudmesh/bin
cd ~/cloudmesh/bin
```

Then download Spark 2.3.2 as follows.

```
wget https://archive.apache.org/dist/spark/spark-2.3.2/spark-2.3.2-bin-without-hadoop.tgz
```

Now extract the file,

```
tar xzf spark-2.3.2-bin-without-hadoop.tgz
```

Then add the environmental variables,

If you have already installed Spark with Hadoop by following section [1.3](#) please update the current SPARK HOME variable with the new path.

```
emacs ~/.bashrc
```

Go to the last line and add the following content.

```
export SPARK_HOME=~/cloudmesh/bin/spark-2.3.2-bin-without-hadoop  
export PATH=$SPARK_HOME/bin:$PATH
```

Source the bashrc file.

```
source ~/.bashrc
```

12.3.2.7 Configuring Hadoop

Now we must add the current Hadoop version that we are using for Spark. Open up a new terminal and then run the following.

```
cd $SPARK_HOME  
  
cd conf  
cp spark-env.sh.template spark-env.sh
```

Now we need to add a new line to show the current path to hadoop installation. Add the following variable in to the spark-env.sh file.

```
emacs spark-env.sh  
export SPARK_DIST_CLASSPATH=$(($HADOOP_HOME/bin/hadoop classpath))
```

Resource	Source
/home/vibhatha/cloudmesh/bin/hadoop-3.0.0/etc/hadoop/	System Classpath
/home/vibhatha/cloudmesh/bin/hadoop-3.0.0/share/hadoop/common/hadoop-common-3.0.0-tests.jar	System Classpath
/home/vibhatha/cloudmesh/bin/hadoop-3.0.0/share/hadoop/common/hadoop-common-3.0.0.jar	System Classpath

Spark Web UI - Hadoop Path

12.3.2.8 Test Spark Installation

Open up a new terminal and then run the following command.

```
spark-shell
```

If it has been configured properly, it will display the following content.

```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLe
Spark context Web UI available at http://149-160-230-133.dhcp-bl.indiana.edu:4040
Spark context available as 'sc' (master = local[*], app id = local-1521732740077).
Spark session available as 'spark'.
Welcome to

    / \
   / \ / \
  / \ / \ / \
 / \ / \ / \ / \
/ \ / \ / \ / \
version 2.3.2

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_151)
Type in expressions to have them evaluated.
Type :help for more information.
```

Then take a look the following address in the browser.

<http://localhost:4040>

Please check the console LOGS and find the port number on which the Spark Web UI is hosted. It will show something like: Spark context Web UI available at the `logs` folder

12.3.3 SPARK STREAMING



12.3.3.1 Streaming Concepts

Spark Streaming is one of the components extending from Core Spark. Spark streaming provides a scalable fault tolerant system with high throughput. For streaming data into spark, there are many libraries like Kafka, Flume, Kinesis, etc.

12.3.3.2 Simple Streaming Example

In this section, we are going to focus on making a simple streaming application using the network in your computer. Here we are going to expose a particular port and from that port we are going to continuously stream data by user entries and the word count is being calculated as the output.

First, create a Makefile

```
mkdir -p ~/cloudmesh/spark/examples/streaming  
cd ~/cloudmesh/spark/examples/streaming  
emacs Makefile
```

Then add the following content to Makefile.

Please add a tab when adding the corresponding command for a given instruction in Makefile. In pdf mode the tab is not clearly shown.

```
SPARKHOME = ${SPARK_HOME}  
run-streaming:  
    ${SPARKHOME}/bin/spark-submit streaming.py localhost 9999
```

Now we need to create file called streaming.py

```
emacs streaming.py
```

Then add the following content.

```
from pyspark import SparkContext  
from pyspark.streaming import StreamingContext  
  
# Create a local StreamingContext with two working thread and batch interval of 1 second  
sc = SparkContext("local[2]", "NetworkWordCount")  
  
log4jLogger = sc._jvm.org.apache.log4j  
LOGGER = log4jLogger.LogManager.getLogger(__name__)  
LOGGER.info("Pyspark script logger initialized")  
  
ssc = StreamingContext(sc, 1)  
  
# Create a DStream that will connect to hostname:port, like localhost:9999  
lines = ssc.socketTextStream("localhost", 9999)  
# Split each line into words  
words = lines.flatMap(lambda line: line.split(" "))  
# Count each word in each batch  
pairs = words.map(lambda word: (word, 1))  
wordCounts = pairs.reduceByKey(lambda x, y: x + y)  
  
# Print the first ten elements of each RDD generated in this DStream to the console  
wordCounts.pprint()  
ssc.start()          # Start the computation  
ssc.awaitTermination(100) # Wait for the computation to terminate
```

To run the code, we need to open up two terminals.

Terminal 1 :

First use netstat to open up a port to start communication.

```
nc -l k 9999
```

Terminal 2 :

Now run the Spark programme in the second terminal.

```
make run-streaming
```

In this terminal you can see an script running trying to read the stream coming from the port 9999. You can enter texts in the Terminal 1 and these texts will be tokenized and the word count is calculated and the result is shown in the Terminal 2.

12.3.3.3 Spark Streaming For Twitter Data

In this section we are going to learn how to use Twitter data as the streaming data source and use Spark Streaming capabilities to process the data. As the first step you must install the python packages using pip.

12.3.3.1 Step 1

```
sudo pip install tweepy
```

12.3.3.2 Step 2

Then you need to create an account in Twitter Apps. Go to and sign in to your twitter account or create a new twitter account. Then you need to create a new application, let's name this application as Cloudmesh-Spark-Streaming.

First you need to create an app with the app name we suggested in this section. The way to create the app is mentioned in +[Figure 65](#).

Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement

Yes, I have read and agree to the [Twitter Developer Agreement](#).

[Create your Twitter application](#)

Figure 65: Create Twitter App

Next we need to take a look at the dashboard created for the app. You can see how your dashboard looks like in +[Figure 66](#).

Your application has been created. Please take a moment to review and adjust your application's settings.

Cloudmesh-Spark-Streaming

[Test OAuth](#)

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

 Testing Spark Streaming
<https://www.vibhatha.org>

Organization
Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Figure 66: Go To Twitter App Dashboard

Next the application tokens generated must be reviewed and it can be found in +[Figure 67](#), you need to go to the [Keys and Access Tokens](#) tab.

The screenshot shows the 'Cloudmesh-Spark-Streaming' application settings page on the Twitter developer platform. At the top, there are tabs for 'Details', 'Settings', 'Keys and Access Tokens' (which is selected), and 'Permissions'. A 'Test OAuth' button is located in the top right corner. Below the tabs, the 'Application Settings' section contains fields for 'Consumer Key (API Key)', 'Consumer Secret (API Secret)', 'Access Level' (set to 'Read and write'), 'Owner', and 'Owner ID'. A note below the consumer key says: 'Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.' Below this is the 'Application Actions' section with buttons for 'Regenerate Consumer Key and Secret' and 'Change App Permissions'. A horizontal scrollbar is visible at the bottom of this section. The 'Your Access Token' section follows, with a note: 'You haven't authorized this application for your own account yet.' It includes a note about generating tokens: 'By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.' Finally, the 'Token Actions' section features a single button labeled 'Create my access token'.

Figure 67: Create Your Twitter Settings

Now you need to generate the access tokens for the first time if you have not generated access tokens and this can be done by clicking the `Create my access token` button. See +[Figure 68](#)

Cloudmesh-Spark-Streaming

Test OAuth

Details Settings **Keys and Access Tokens** Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)

Consumer Secret (API Secret)

Access Level Read and write ([modify app permissions](#))

Owner

Owner ID

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

[Create my access token](#)

Figure 68: Create Your Twitter Access Tokens

The access tokens and keys are blurred in this section for privacy issues.

12.3.3.3 Step 3

Let us build a simple Twitter App to see if everything is okay.

```
mkdir -p ~/cloudmesh/spark/streaming
cd ~/cloudmesh/spark/streaming
emacs twitterstreaming.py
```

Add the following content to the file and make sure you update the corresponding token keys with your token values.

```
import tweepy
```

```

CONSUMER_KEY = 'your_consumer_key'
CONSUMER_SECRET = 'your_consumer_secret'
ACCESS_TOKEN = 'your_access_token'
ACCESS_TOKEN_SECRET = 'your_access_token_secret'

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth)

status = "Testing!"
api.update_status(status=status)

```

python twitterstreaming.py

12.3.3.4 Step 4

Let us start the twitter streaming exercise. We need to create a Tweet Listener in order to retrieve data from twitter regarding a topic of your choice. In this exercise, we have tried keywords like `trump, indiana, messi.`

```

mkdir -p ~/cloudmesh/spark/streaming
cd ~/cloudmesh/spark/streaming
emacs tweetlistener.py

```

Make your to replace strings related to secret keys and ip addresses by replacing these values depending on your machine configuration and twitter keys.

Now add the following content.

```

import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json

CONSUMER_KEY = 'YOUR_CONSUMER_KEY'
CONSUMER_SECRET = 'YOUR_CONSUMER_SECRET'
ACCESS_TOKEN = 'YOUR_ACCESS_TOKEN'
ACCESS_SECRET = 'YOUR_SECRET_ACCESS'

class TweetListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:

```

```

        msg = json.loads( data )
        print( msg['text'].encode('utf-8') )
        self.client_socket.send( msg['text'].encode('utf-8') )
        return True
    except BaseException as e:
        print("Error on_data: %s" % str(e))
    return True

    def on_error(self, status):
        print(status)
        return True

    def sendData(c_socket):
        auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
        auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

        twitter_stream = Stream(auth, TweetListener(c_socket))
        twitter_stream.filter(track=['messi']) # you can change this topic

    if __name__ == "__main__":
        s = socket.socket()
        host = "YOUR_MACHINE_IP"
        port = 5555
        s.bind((host, port))

        print("Listening on port: %s" % str(port))

        s.listen(5)
        c, addr = s.accept()

        print( "Received request from: " + str( addr ) )

        sendData( c )

```

12.3.3.5 step 5

Please replace the local file paths mentioned in this code with a file path of your preference depending on your workstation. And also IP address must be replaced with your ip address. The log folder path must be pre-created and make sure to replace the `registerTempTable` name with respect to the entity that you are referring. This will minimize the conflicts among different topics when you need to plot it in a simple manner.

Add the following content to the IpythonNote book as follows

Open up a terminal,

```

cd ~/cloudmesh/spark/streaming
jupyter notebook

```

Then in the browser the jupyter notebook is being loaded. There create a new IPython notebook called twittersparkstremer.

Then add the following content.

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import SQLContext
from pyspark.sql.functions import desc

sc = SparkContext('local[2]', 'twittersparkstreamer')

ssc = StreamingContext(sc, 10 )
sqlContext = SQLContext(sc)
ssc.checkpoint( "file:///home/<your-username>/cloudmesh/spark/streaming/logs/messi")

socket_stream = ssc.socketTextStream("YOUR_IP_ADDRESS", 5555)

lines = socket_stream.window( 20 )

from collections import namedtuple
fields = ("tag", "count")
Tweet = namedtuple( 'Tweet', fields )

( lines.flatMap( lambda text: text.split( " " ) )
    .filter( lambda word: word.lower().startswith("#") )
    .map( lambda word: ( word.lower(), 1 ) )
    .reduceByKey( lambda a, b: a + b )
    .map( lambda rec: Tweet( rec[0], rec[1] ) )
    .foreachRDD( lambda rdd: rdd.toDF().sort( desc("count") )
        .limit(10).registerTempTable("tweetsmessi") ) )#change table name depend:

sqlContext

<pyspark.sql.context.SQLContext at 0x7f51922ba350>

ssc.start()

import matplotlib.pyplot as plt
import seaborn as sn

import time
from IPython import display

count = 0
while count < 10:
    time.sleep( 20 )
    top_10_tweets = sqlContext.sql( 'Select tag, count from tweetsmessi' ) #change table
    top_10_df = top_10_tweets.toPandas()
    display.clear_output(wait=True)
    #sn.figure( figsize = ( 10, 8 ) )
    sn.barplot( x="count", y="tag", data=top_10_df)
    plt.show()
```

```
count = count + 1  
  
ssc.stop()
```

12.3.3.3.6 step 6

Open `Terminal 1`, then do the following

```
cd ~/cloudmesh/spark/streaming  
python tweetslistener.py
```

It will show that:

```
Listening on port: 5555
```

Open `Terminal 2`

Now we must start the Spark app by running the content in the IPython Notebook by pressing `SHIFT-ENTER` in each box to run each command. Make sure not to run twice the starting command of the `SparkContext` or initialization of `SparkContext`.

Now you will see streams in the `Terminal 1` and you can see plots after a while in the IPython Notebook.

Sample outputs can be seen in [+Figure 69](#), [+Figure 70](#), [+Figure 71](#), [+Figure 72](#).

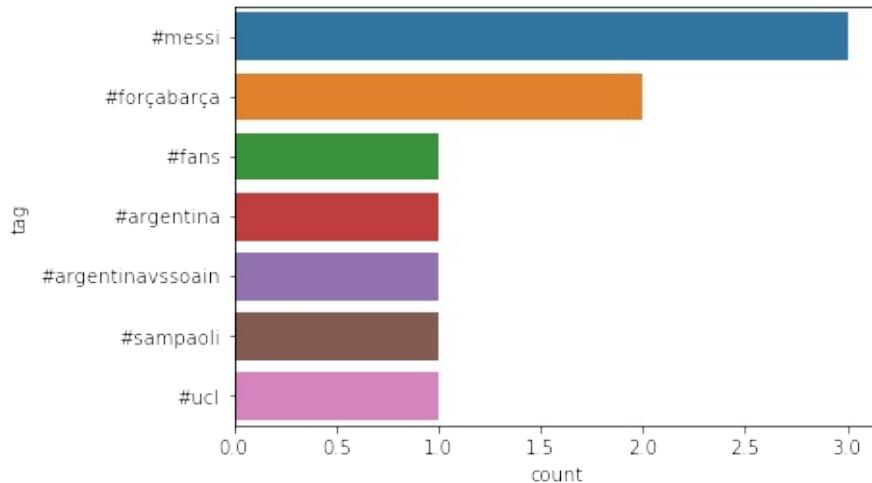


Figure 69: Twitter Topic Messi

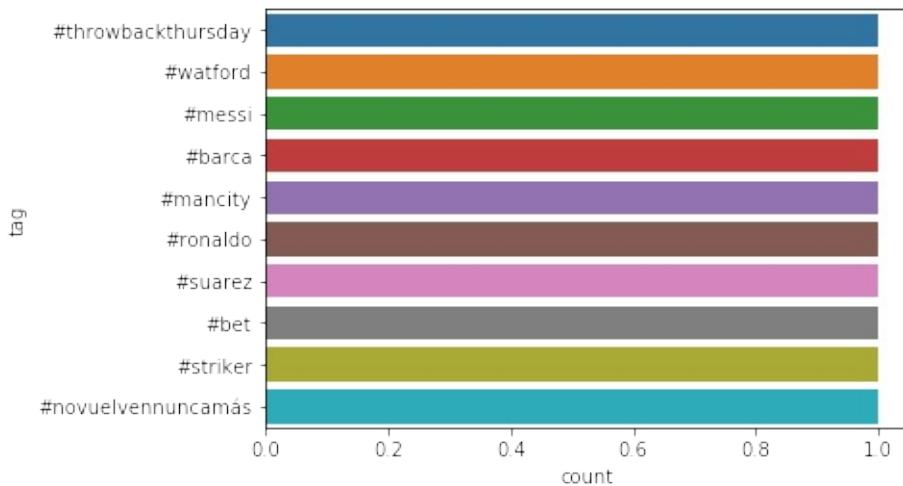


Figure 70: Twitter Topic Messi

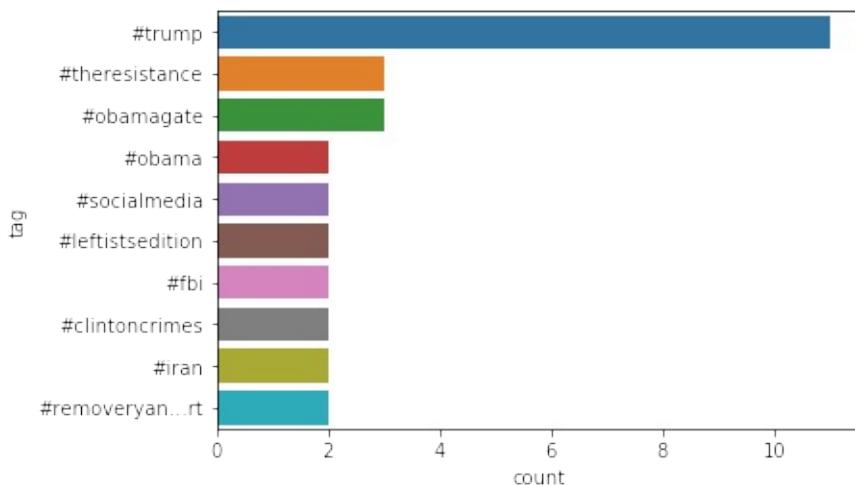


Figure 71: Twitter Topic Messi

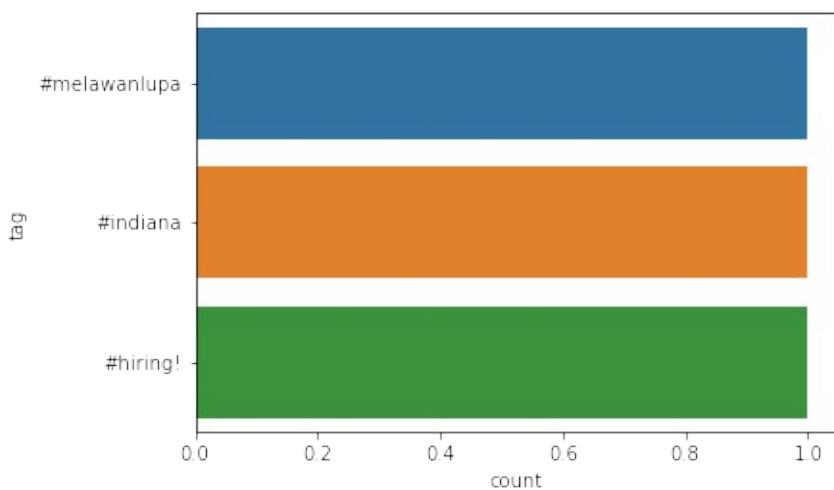


Figure 72: Twitter Topic Messi

12.3.4 USER DEFINED FUNCTIONS IN SPARK



Apache Spark is a fast and general cluster-computing framework which perform computational tasks up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk for high speed large-scale streaming, machine learning and SQL workloads tasks. Spark offers support for the applications development employing over 80 high-level operators using Java, Scala, Python, and R. Spark powers the combined or standalone use of a stack of libraries including SQL and DataFrames, MLLib for machine learning, GraphX, and Spark Streaming. Spark can be utilized in standalone cluster mode, on EC2, on Hadoop YARN, or on Apache Mesos and it allows data access in HDFS, Cassandra, HBase, Hive, Tachyon, and any Hadoop data source.

User-defined functions (UDFs) are the functions created by developers when the built-in functionalities offered in a programming language, are not sufficient to do the required work. Similarly, Apache Spark UDFs also allow developers to enable new functions in higher level programming languages by extending built-in functionalities. It also allows developers to experiment with wide range of options for integrating UDFs with Spark SQL, MLib and GraphX workflows.

This tutorial explains following:

How to install Spark in Linux, Windows and MacOS.

How to create and utilize user defined functions(UDF) in Spark using Python.

How to run the provided example using a provided docker file and make file.

12.3.4.1 Resources

- <https://spark.apache.org/>
- <http://www.scala-lang.org/>
- <https://docs.databricks.com/spark/latest/spark-sql/udf-in-python.html>

12.3.4.2 Instructions for Spark installation

12.3.4.2.1 Linux

First, JDK (Recommended version 8) should be installed to a path where there is no space.

- <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Second, setup environment variables for JDK by adding bin folder path to user path variable.

```
This $ export PATH = $PATH:/usr/local/java8/bin
```

Next, download and extract Scala pre-built version from

- <http://www.scala-lang.org/download/>

Then, setup environment variables for Scala by adding bin folder path to the user path variable.

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

Next, download and extract Apache Spark pre-built version.

- <https://spark.apache.org/downloads.html>

Then, setup environment variables for spark by adding bin folder path to the user path variable.

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

Finally, for testing the installation, please type the following command.

```
spark-shell
```

12.3.4.3 Windows

First, JDK should be installed to a path where there is no space in that path. Recommended JAVA version is 8.

<http://www.oracle.com/technetwork/java/javase/downloads/index.htm>

Second, setup environment variables for jdk by adding bin folder path to to user path variable.

```
set JAVA_HOME=c:\java8  
set PATH=%JAVA_HOME%\bin;%PATH%
```

Next, download and extract Apache Spark pre-built version.

<https://spark.apache.org/downloads.html>

Then, setup environment varibale for spark by adding bin folder path to the user path variable.

```
set SPARK_HOME=c:\spark  
set PATH=%SPARK_HOME%\bin;%PATH%
```

Next, download the winutils.exe binary and Save winutils.exe binary to a directory (`c:\hadoop\bin`).

<https://github.com/steveloughran/winutils>

Then, change the winutils.exe permission using following command using CMD with administrator permission.

```
$ winutils.exe chmod -R 777 C:\tmp\hive
```

If your system doesnt have `hive` folder, make sure to create `C:\tmp\hive` directory.

Next, setup environment variables for hadoop by adding bin folder path to the user path variable.

```
set HADOOP_HOME=c:\hadoop\bin  
set PATH=%HADOOP_HOME%\bin;%PATH%
```

Then, install Python 3.6 with anaconda (This is a bundled python installer for pyspark).

<https://anaconda.org/anaconda/python>

Finally, for testing the installation, please type the following command.

```
$ pyspark
```

12.3.4.4 MacOS

First, JDK should be installed to a path where there is no space in that path. Recommended JAVA version is 8.

<http://www.oracle.com/technetwork/java/javase/downloads/index.htm>

Second, setup environment variables for jdk by adding bin folder path to user path variable.

```
$ export JAVA_HOME=$(/usr/libexec/java_home)
```

Next, Install Apache Spark using Homebrew with following commands.

```
$ brew update  
$ brew install scala  
$ brew install apache-spark
```

Then, setup environment variable for spark with following commands.

```
$ export SPARK_HOME="/usr/local/Cellar/apache-spark/2.1.0/libexec/"  
$ export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/build:$PYTHONPATH  
$ export PYTHONPATH=$SPARK_HOME/python/lib/py4j-0.10.4-src.zip:$PYTHONPATH
```

Next, install Python 3.6 with anaconda (This is a bundled python installer for pyspark)

<https://anaconda.org/anaconda/python>

Finally, for testing the installation, please type the following

command.

```
$ pyspark
```

12.3.4.5 Instructions for creating Spark User Defined Functions

12.3.4.5.1 Example: Temperature conversion

In this example we convert temperature data from Celsius to Fahrenheit with filtering and sorting

12.3.4.5.1.1 Description about data set

The file **temperature_data.csv** contains temperature data of different wheather stations and it has the following structure.

```
ITE00100554,18000101,TMAX,-75,,,E,  
ITE00100554,18000101,TMIN,-148,,,E,  
GM000010962,18000101,PRCP,0,,,E,  
EZE00100082,18000101,TMAX,-86,,,E,  
GM000010962,18000104,PRCP,0,,,E,  
EZE00100082,18000104,TMAX,-55,,,E,
```

We will only consider wheather station ID (column 0), entrytype (column 2), temperature (column 3: it is in 10*Celsius)

12.3.4.5.1.2 How to write a python program with UDF

First, we need to import the relevant libraries to use Spark sql built in functionalities listed as follows.

```
from pyspark.sql import SparkSession  
from pyspark.sql import Row
```

Then, we need create a user defined fuction which will read the text input and process the data and return a spark sql Row object. It can be created as listed as follows.

```
def process_data(line):  
    fields = line.split(',')  
    stationID = fields[0]  
    entryType = fields[2]  
    temperature = float(fields[3]) * 0.1 * (9.0 / 5.0) + 32.0  
    return Row(ID=stationID, t_type=entryType, temp=temperature)
```

Then we need to create a Spark SQL session as listed as follows with an application name.

```
spark = SparkSession.builder.appName("Simple SparkSQL UDF example").getOrCreate()
```

Next, we read the raw data using spark build-in function `textFile()` as shown next.

```
lines = spark.sparkContext.textFile("temperature_data.csv")
```

Then, we convert those read lines to a Resilient Distributed Dataset (RDD) of Row object using UDF (`process_data`) which we created as listed as follows.

```
parsedLines = lines.map(process_data)
```

Alternatively we could have written the UDF using a python lamda function to do the same thing as shown next.

```
parsedLines = lines.map(lambda line: Row(ID=line.split(',')[0],  
                                         t_type=line.split(',')[2],  
                                         temp=float(line.split(',')[3]) * 0.1 * (9.0  
                                         / 5.0) + 32.0))
```

Now, we can convert our RDD object to a Spark SQL Dataframe as listed as follows.

```
TempDataset = spark.createDataFrame(parsedLines)
```

Next, we can print and see the first 20 rows of data to validate our work as shown next.

```
TempDataset.show()
```

12.3.4.5.1.3 How to execute a python spark script

You can use **spark-submit** command to run a spark script as shown next.

```
spark-submit temperature_converter.py
```

If everything went well, you should see the following output.

```
+-----+-----+-----+
```

ID	t_type	temp
EZE00100082	TMAX	90.14000000000001
ITE00100554	TMAX	90.14000000000001
ITE00100554	TMAX	89.42
EZE00100082	TMAX	88.88
ITE00100554	TMAX	88.34
ITE00100554	TMAX	87.80000000000001
ITE00100554	TMAX	87.62
ITE00100554	TMAX	87.62
EZE00100082	TMAX	87.26
EZE00100082	TMAX	87.08000000000001
EZE00100082	TMAX	87.08000000000001
ITE00100554	TMAX	86.72
ITE00100554	TMAX	86.72
ITE00100554	TMAX	86.72
EZE00100082	TMAX	86.72
ITE00100554	TMAX	86.0
ITE00100554	TMAX	86.0
ITE00100554	TMAX	86.0
ITE00100554	TMAX	85.64
ITE00100554	TMAX	85.64

only showing top 20 rows

12.3.4.5.1.4 Filtering and sorting

Now we are trying to find what is the maximum temperature reported for a particular weather station and print the data in ascending order. We can achieve this by using **where()** and **orderBy()** functions as shown next.

```
TempDatasetProcessed = TempDataset.where(TempDataset['t_type'] == 'TMAX')
    .orderBy('temp', ascending=False).cache()
```

We achieved the filtering using temperature type and it filters out all the data which is not a TMAX.

Finally, we can print the data to see whether this worked or not using following statement.

```
TempDatasetProcessed.show()
```

Now, it is the time to run the python script again using following command.

```
spark-submit temperature_converter.py
```

If everything went well, you should see the following sorted and

filtered output.

ID	t_type	temp
EZE00100082	TMAX	90.14000000000001
ITE00100554	TMAX	90.14000000000001
ITE00100554	TMAX	89.42
EZE00100082	TMAX	88.88
ITE00100554	TMAX	88.34
ITE00100554	TMAX	87.80000000000001
ITE00100554	TMAX	87.62
ITE00100554	TMAX	87.62
EZE00100082	TMAX	87.26
EZE00100082	TMAX	87.08000000000001
EZE00100082	TMAX	87.08000000000001
ITE00100554	TMAX	86.72
ITE00100554	TMAX	86.72
ITE00100554	TMAX	86.72
EZE00100082	TMAX	86.72
ITE00100554	TMAX	86.0
ITE00100554	TMAX	86.0
ITE00100554	TMAX	86.0
ITE00100554	TMAX	85.64
ITE00100554	TMAX	85.64

Complete python script is listed as follows as well as under this directory (temperature_converter.py).

https://github.com/cloudmesh-community/hid-sp18-409/blob/master/tutorial/spark_udfs/temperature_converter.py

```
from pyspark.sql import SparkSession
from pyspark.sql import Row

def process_data(line):
    fields = line.split(',')
    stationID = fields[0]
    entryType = fields[2]
    temperature = float(fields[3]) * 0.1 * (9.0 / 5.0) + 32.0
    return Row(ID=stationID, t_type=entryType, temp=temperature)

# Create a SparkSQL Session
spark = SparkSession.builder.appName('Simple SparkSQL UDF example'
    ).getOrCreate()

# Get the raw data
lines = spark.sparkContext.textFile('temperature_data.csv')

# Convert it to a RDD of Row objects
parsedLines = lines.map(process_data)
```

```

# alternative lambda function

parsedLines = lines.map(lambda line: Row(ID=line.split(',')[0],
                                         t_type=line.split(',')[2],
                                         temp=float(line.split(',')[3]) * 0.1 * (9.0
                                         / 5.0) + 32.0))

# Convert that to a DataFrame
TempDataset = spark.createDataFrame(parsedLines)

# show first 20 rows temperature converted data
# TempDataset.show()

# Some SQL-style magic to sort all movies by popularity in one line!
TempDatasetProcessed = TempDataset.where(TempDataset['t_type'] == 'TMAX'
                                         ).orderBy('temp', ascending=False).cache()

# show first 20 rows of filtered and sorted data
TempDatasetProcessed.show()

```

12.3.4.6 Instructions to install and run the example using docker

Following link is the home directory for the example explained in this tutorial.

https://github.com/cloudmesh-community/hid-sp18-409/tree/master/tutorial/spark_udfs

It contains following files

- Python script which contains the example: [temperature_converter.py](#)
- Temperature data file: [temperature_data.csv](#)
- Required python dependencies are put here: [requirements.txt](#)
- Docker file which automatically setup the codebase with dependency installation: [Dockerfile](#)
- Make file which will excute the example with a single command: [Makefile](#)

To install the example using docker plesse do the following steps.

First, you should install docker in to your computer.

Next, git clone the [project](#). Alternatively you can also download the docker image from the docker hub. Then you don't need to do docker build.

```
$ docker pull kadupitiya/tutorial
```

Then, change the directory to **spark_udfs** folder.

Next, install the service using following make command

```
$ make docker-build
```

Finally, start the service using following make command

```
$ make docker-start
```

Now you should see the same output we saw at the end of the example explanation.

12.4 ADVANCED MAPREDUCE TOPICS



12.4.1 AMAZON EMR (ELASTIC MAP REDUCE)



Amazon EMR provides a managed Hadoop framework that makes big data processing - Easy - Fast - Cost-effective

EMR Supports other distributed framework such as Apache Spark, HBase, Presto, Flink and etc. Interact with data in AWS data stores such as Amazon S3, DynamoDB and etc.

Components Of EMR: - Storage - EC2 instance - Clusters

12.4.1.1 Why EMR?

Easy to Use - Launch cluster in a min

Pay as you go - Pay an hourly rate

Flexible - Easily Add/ Remove capacity

Reliable - Spend less time for monitoring

Secure - Manage firewall

12.4.1.2 AWS Storage

S3 - Cloud based storage - Accessible from anywhere

Instance Store - Local storage - Data will be lost on start and stop EC2 instances

EBS - Network attached storage - Data preserved on start and stop - Accessible only through EC2 instances

12.4.1.3 Create EMR in AWS

12.4.1.3.1 Create the buckets

- Login to AWS console and create the buckets at <https://aws.amazon.com/console/>. To create the buckets, go to services (see [Figure 73](#), [Figure 74](#)), click on S3 under Storage, [Figure 75](#), [Figure 76](#), [Figure 77](#). Click on Create bucket button and then provide all the details to complete bucket creation.
- AWS Console



Figure 73: AWS Console

- AWS Login

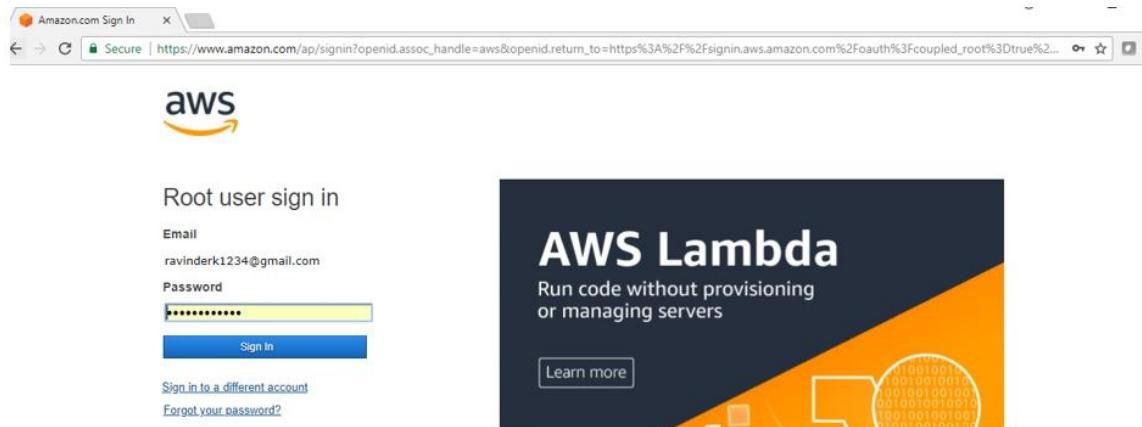


Figure 74: AWS Login

- S3 – Amazon Storage

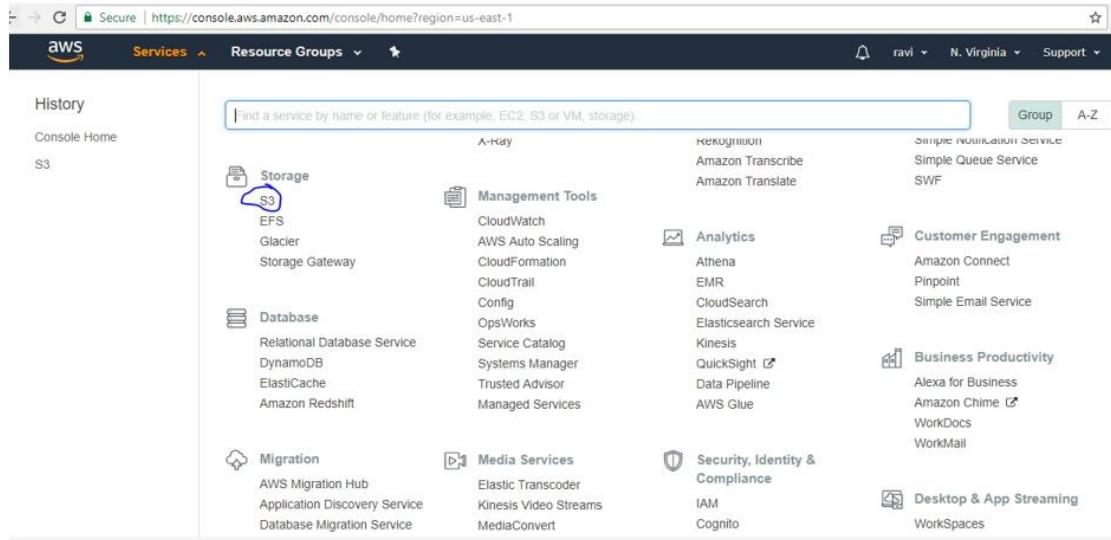


Figure 75: Amazon Storage

- S3 – Create buckets

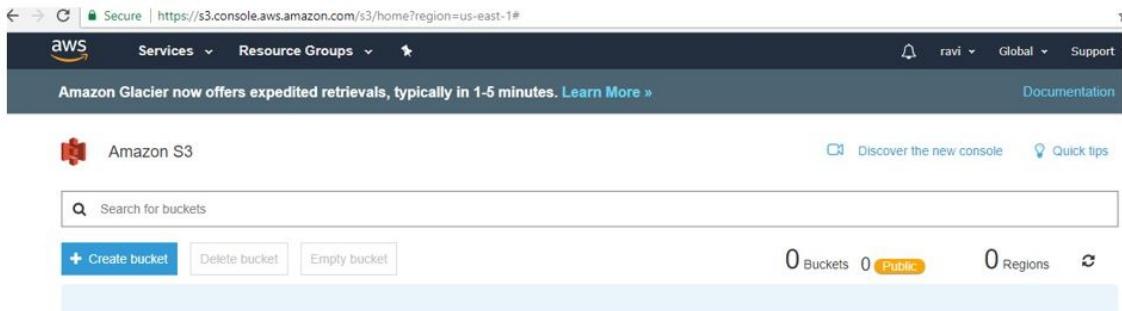


Figure 76: S3 buckets

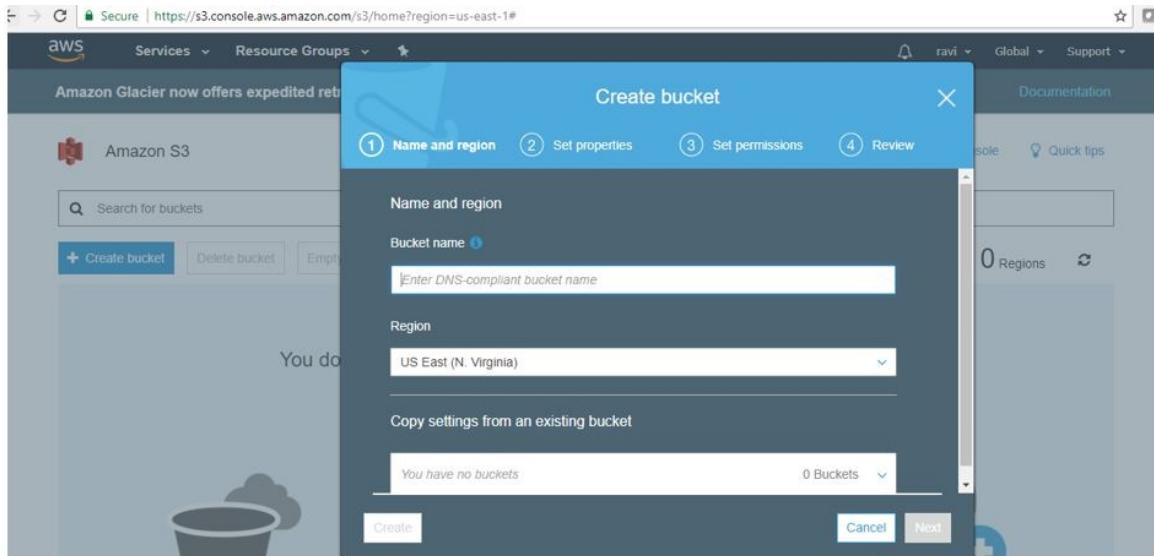


Figure 77: S3 buckets1

12.4.1.3.2 Create Key Pairs

- Login to AWS console, go to services, click on EC2 under compute. Select the Key pairs resource, click on Create Key Pair and provide Key Pair name to complete the Key pairs creation. See [Figure 78](#)
- Download the. pem file once Key value pair is created. This is needed to access AWS Hadoop environment from client machine. This need to be imported in Putty to access your AWS environemnt. See [Figure 79](#)

12.4.1.3.2.1 Create Key Value Pair Screen shots

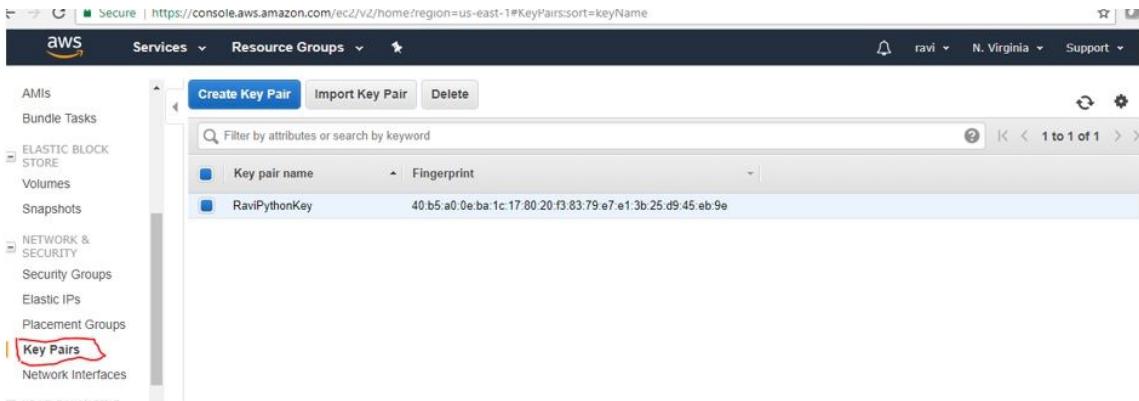


Figure 78: AMS Key Value Pair

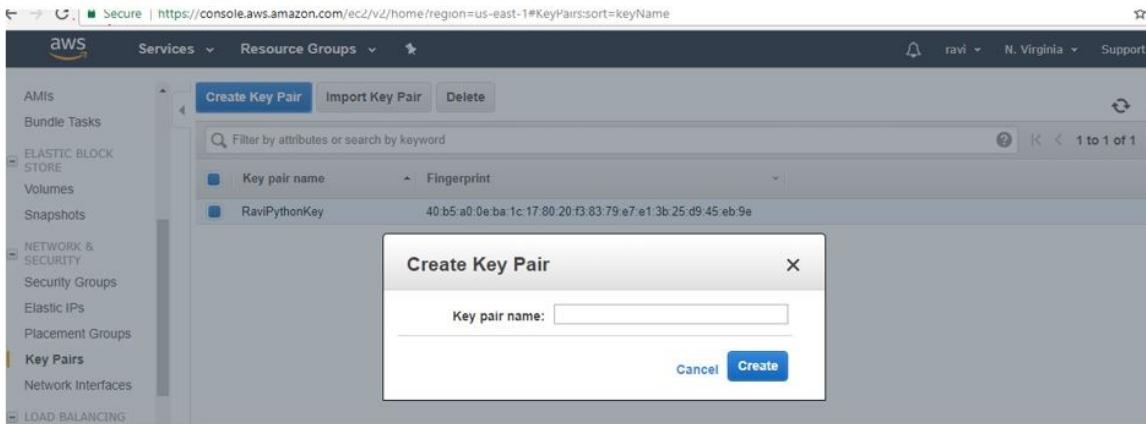


Figure 79: AMS Key Value Pair1

12.4.1.4 Create Step Execution – Hadoop Job

Login to AWS console, go to services and then select EMR. Click on Create Cluster. In the cluster configuration provide below details to complete to complete step execution creation. See: [Figure 80](#), [Figure 81](#), [Figure 82](#), [Figure 83](#), [Figure 84](#) - Cluster name (Ex: HadoopJobStepExecutionCluster) - Select Logging check box and provide S3 folder location (Ex: s3://bigdata-raviAndOrlyiuproject/logs/) - Select launch mode as Step execution - Select the step type and complete the step configuration - Complete Software Configuration - Complete Hardware Configuration - Complete Security and access - And then click on create cluster button - Once job started, if there are no errors output file will be created in the output directory.

12.4.1.4.0.1 Screen shots

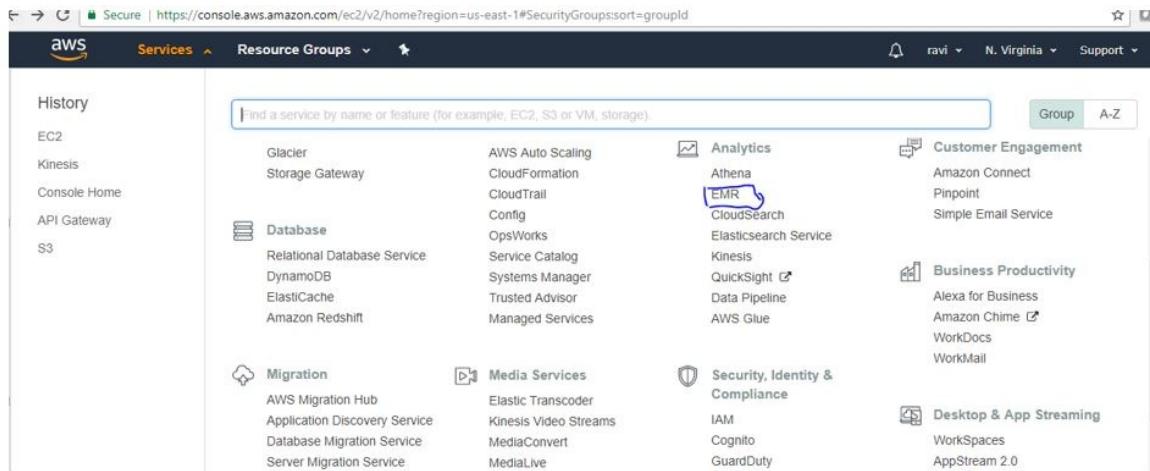


Figure 80: AWS EMR

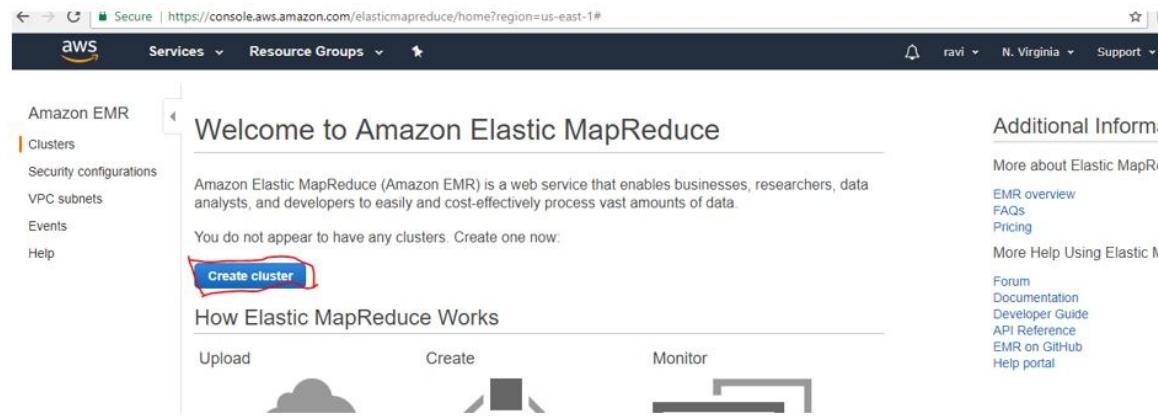


Figure 81: AWS Create EMR

Create Cluster - Quick Options [Go to advanced options](#)

General Configuration

Cluster name [?](#)

Logging [?](#)

S3 folder [?](#)

Launch mode Cluster [?](#) Step execution [?](#)

Add steps

A step is a unit of work submitted to an application running on your EMR cluster. EMR programmatically installs the applications needed to execute the added steps. [Learn more](#)

Step type [Configure](#)

Software configuration

Release [?](#)

Applications Hadoop 2.8.3 [?](#)

Figure 82: AWS Config EMR

Security and access

Permissions Default Custom
Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role [?](#)

EC2 instance profile [?](#)

[Cancel](#) [Create cluster](#)

Figure 83: AWS Create Cluster

Figure 84: AWS Create Cluster1

12.4.1.5 Create a Hive Cluster

Login to AWS console, go to services and then select EMR. Click on Create Cluster. In the cluster configuration provide below details to complete. See, [Figure 85](#), [Figure 86](#), [Figure 87](#) - Cluster name (Ex: MyFirstCluster-Hive) - Select Logging check box selected and provide S3 folder location - Select launch mode as Cluster - Complete software configuration (select hive application) and click on create cluster - ### Create a Hive Cluster - Screen shots

Figure 85: Hive Cluser

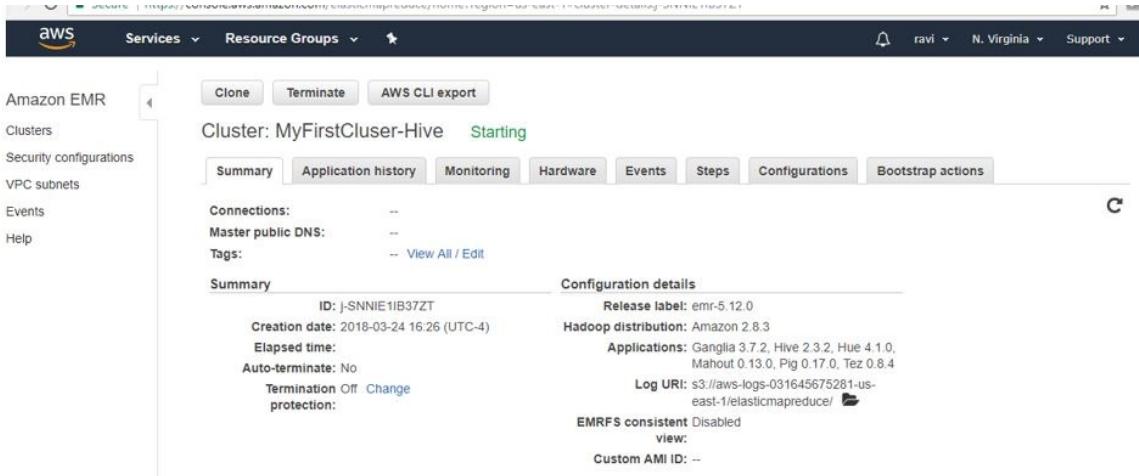


Figure 86: Hive Cluser1

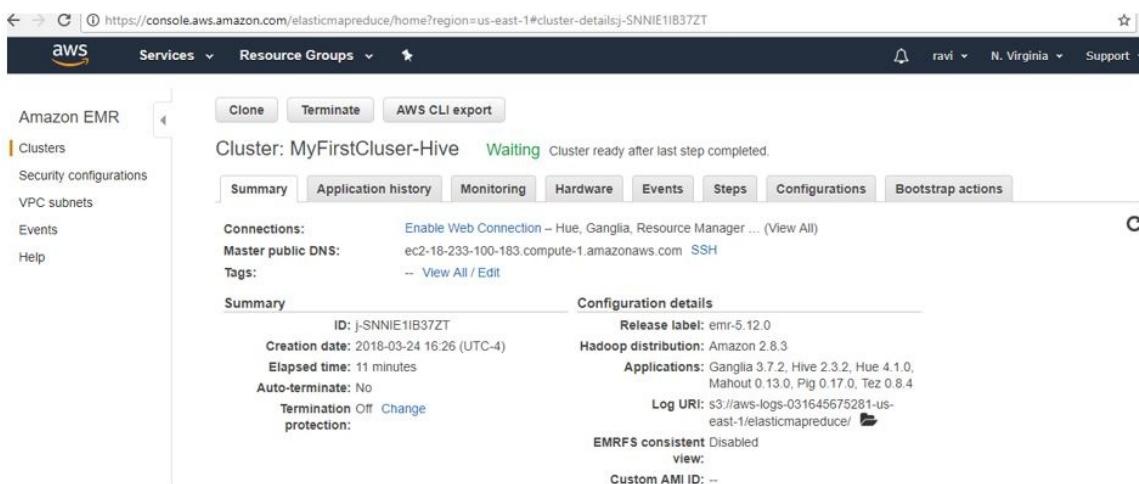


Figure 87: Hive Cluser2

12.4.1.6 Create a Spark Cluster

Login to AWS console, go to services and then select EMR. Click on Create Cluster. In the cluster configuration provide below details to complete. See, [Figure 88](#), [Figure 89](#), [Figure 90](#) - Cluster name (Ex:My Cluster - Spark) - Select Logging check box selected and provide S3 folder location - Select launch mode as Cluster - Complete software configuration and click on create cluster - Select application as Spark

12.4.1.6.1 Create a Spark Cluster - Screenshots

The screenshot shows the 'Create Cluster - Quick Options' page on the AWS console. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', and user information ('ravi', 'N. Virginia', 'Support'). Below the navigation is the main form area.

General Configuration

- Cluster name:** My cluster
- Logging:** S3 folder `s3://aws-logs-031645675281-us-east-1/elasticmapreduce`
- Launch mode:** Cluster (selected)

Software configuration

- Release:** emr-5.12.0
- Applications:** Core Hadoop: Hadoop 2.8.3 with Ganglia 3.7.2, Hive 2.3.2, Hue 4.1.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4 (radio button selected)
- HBase: HBase 1.4.0 with Ganglia 3.7.2, Hadoop 2.8.3, Hive 2.3.2, Hue 4.1.0, Phoenix 4.13.0, and ZooKeeper 3.4.10
- Presto: Presto 0.188 with Hadoop 2.8.3 HDFS and Hive 2.3.2 Metastore
- Spark:** Spark 2.2.1 on Hadoop 2.8.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3 (radio button selected)

Use AWS Glue Data Catalog for table metadata

Figure 88: Spark Cluser

This screenshot shows the 'Hardware configuration' section of the AWS EMR cluster creation interface.

Instance type: m3.xlarge

Number of instances: 3 (1 master and 2 core nodes)

Figure 89: Spark Cluser

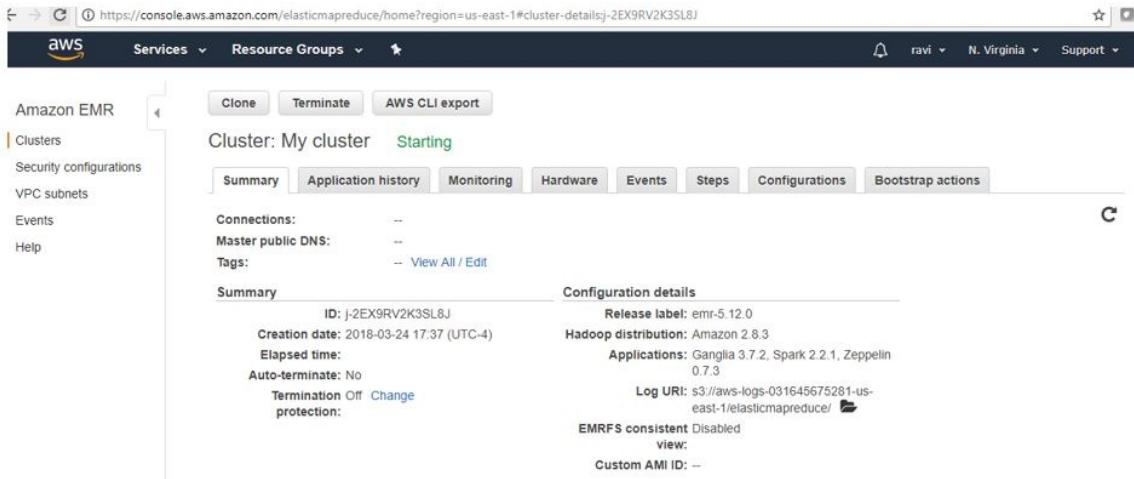


Figure 90: Spark Cluser

12.4.2 AMAZON EMR



Amazon EMR is a Hadoop framework that allows users to process data on the AWS platform using their EC2 technology to spread the load across multiple EC2 instances. Elasticity a major benefit of EMR as it can be set to auto scale up or down the number of EC2 instances that EMR is running in a cluster. The user can choose to run additional frameworks supported on EMR in addition to Hadoop, such as Spark, HBase, Flink and Presto. The platform allows the user to focus on the processing of the data and not have to deal with the setup, management or tuning of a Hadoop cluster. Using EMR allows a user to setup and provision a cluster quickly and allows for scalability of compute resources up or down and in or out as needed. Interactions with EMR can occur through a web service interface or by using the AWS Management Console to launch and monitor clusters. In this section, setup and configuration of an EMR cluster will be done through the AWS Management Console.

12.4.2.1 Prerequisites

Before proceeding with any steps to create an EMR cluster, you need to ensure that you have an AWS account setup. If not, you need to have one created before continuing with this section. Afterwards, sign into the AWS Management Console. There are two prerequisites that

need to be met before being able to launch an EMR cluster, setup an S3 bucket and an EC2 key pair. The S3 bucket you create will be used for storing the EMR logs and any output data produced by your EMR cluster. Bucket names have several constraints that need to be met due to Hadoop requirements such as only consisting of lowercase letters, numbers, periods, hyphens and also cannot end in numbers [37]. In order to create the S3 bucket, go to S3 section of the AWS Management Console and select the ‘Create bucket’ button. Fill in the requested information for bucket name and region, then proceed to the next two pages to enter in the bucket properties and permissions. After those sections, a review page will be presented for your review and if you are satisfied with your selections, click ‘Create bucket’ [38]. As the following: [Figure 91](#).

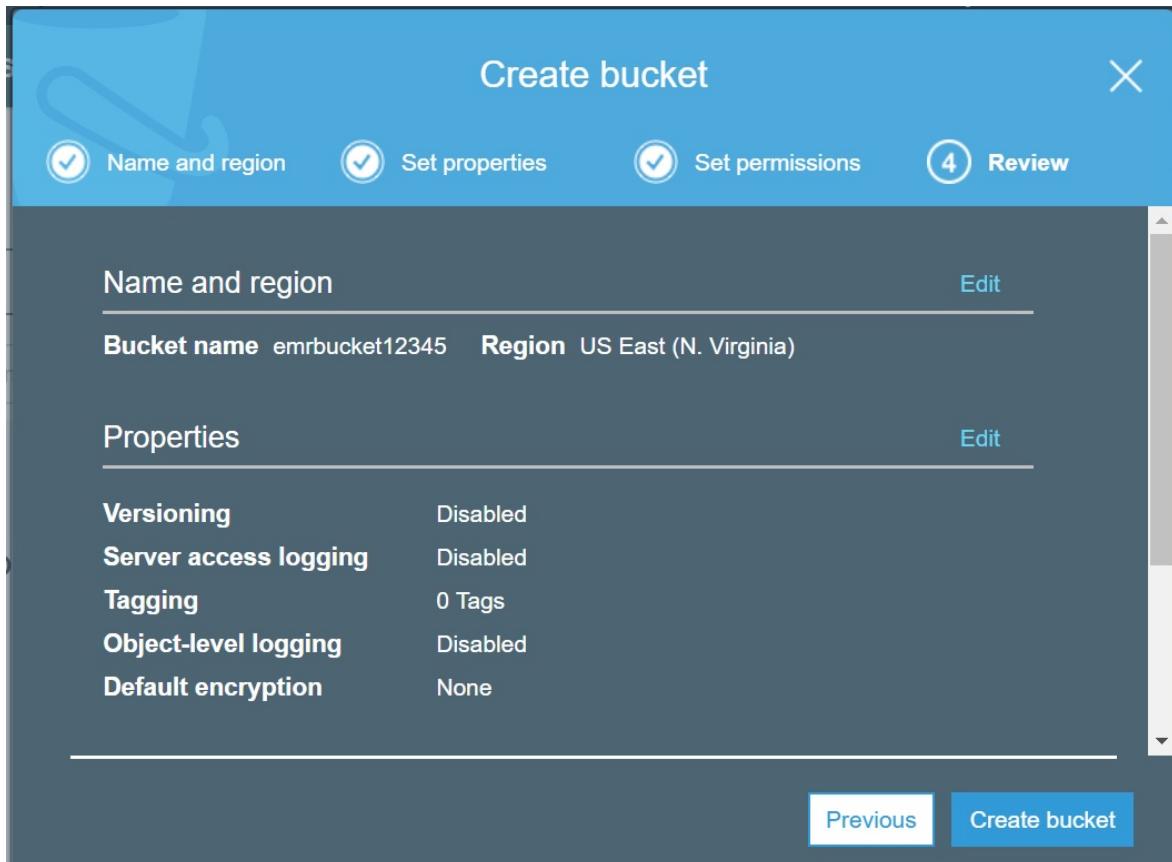


Figure 91: S3 Bucket

After the S3 bucket has been created, an EC2 key pair then needs to be generated which allows you to connect to your EMR cluster over SSH. If you already have an existing EC2 key pair that can be used for

this section as well. In order to generate an EC2 key pair through the AWS Management Console, search and navigate to the EC2 section from the console home page. In the pane on the left side of the screen, look for the ‘Network and Security’ section and select the ‘Key Pairs’ option. On the next page select ‘Create Key Pair’ button and type in a name you want to give this key pair. After you select the key name click ‘Create’ and your EC2 key pair will be automatically downloaded by the browser and named with a pem extension. Save the key file generated in a safe location for use later on as this is the only time you will be able to save this key file. It will need to be used when you need to launch and connect to the EC2 instances created by EMR [39]. As the picture below: [Figure 92](#).

The screenshot shows the AWS Management Console interface for managing EC2 key pairs. At the top, there are three buttons: 'Create Key Pair' (highlighted in blue), 'Import Key Pair', and 'Delete'. Below these is a search bar with a magnifying glass icon and the placeholder text 'Filter by attributes or search by keyword'. The main list area has two columns: 'Key pair name' and 'Fingerprint'. A single item is listed: 'EMR' with the fingerprint 'b3:44:bb:7a:21:e3:d2:57:74:4a:a7:23:16:06:bf:31:d1:21:13:ec'. Below this list is a detailed view for the 'EMR' key pair, showing its 'Key pair name' as 'EMR' and its 'Fingerprint' as 'b3:44:bb:7a:21:e3:d2:57:74:4a:a7:23:16:06:bf:31:d1:21:13:ec'.

Figure 92: Key

12.4.2.2 Configuring and Utilizing the Cluster with AWS Management Console

Once all the prerequisites above have been met, an EMR cluster can now be created. In the AWS Management Console home page, enter in ‘EMR’ in the AWS services search box at the top of the page and select the EMR option that it returns. You will be taken to the EMR home page. At the top of this page is a button labeled ‘Create cluster’. Click this button and you will be brought to the screen where you can select cluster creation options. By default, it brings up the quick

options page which allows you to select basic options needed to setup and configure an EMR cluster. There is also an option to show advanced options that can be used for cluster creation at the top of the page. Stay on the quick options page for now and enter in the cluster name, which is optional. You then select if you would like to enable logging. If so, you can then select a location in S3 where you would like the logs to be placed. There are then two options to choose from for launch mode: cluster or step execution. The cluster option can be used for EMR clusters that you want to remain online indefinitely. The step execution option would be used for when you want to execute a set of predefined steps upon cluster creation and once those steps complete successfully, shut down the cluster [40]. As the figure shows: [Figure 93](#).

General Configuration

The screenshot shows the 'General Configuration' section of the AWS EMR console. It includes fields for 'Cluster name' (set to 'EMRCluster'), 'Logging' (checked), 'S3 folder' (set to 's3://aws-logs-566689635541-us-east-1/elasticmapred'), and 'Launch mode' (set to 'Cluster'). Below the configuration section, there is a note: 'The core Hadoop install is the only application option you can use here.'

Figure 93: Configuration

Once launch mode is selected, scroll down to the 'Software configuration' section and choose an EMR release version. See the following picture: [Figure 94](#). The most current version is selected by default. If you chose a cluster launch mode, then proceed to selecting one of the 4 predefined sets of applications you want to install based on your use case. If step execution was selected instead of cluster, then the core Hadoop install is the only application option you can use here. Now move on to the 'Hardware configuration' and select the EC2 instance type and number of EC2 instances you would like to utilize for your cluster. The values selected here will vary on what type of data processing you're looking to achieve. Then scroll down to the 'Security and access' section of the page and select the key pair generated in the steps above in the drop-down menu, as below: [Figure 95](#). Below that you can then select which permissions model to use: default or custom. The default option sets up permissions for

your EMR cluster that are granted using policies applied to EMR specific IAM roles. Using the custom option allows you to select existing IAM roles to apply permissions to instead of creating new roles. Once all of this information has been selected, the cluster is then ready to be launched by selecting the 'Create cluster' button at the bottom of the page. Your cluster will then be launched and ready for workloads [40].

Software configuration

Release ▼

Applications Core Hadoop: Hadoop 2.8.3 with Ganglia 3.7.2, Hive 2.3.2, Hue 4.1.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4
 HBase: HBase 1.4.0 with Ganglia 3.7.2, Hadoop 2.8.3, Hive 2.3.2, Hue 4.1.0, Phoenix 4.13.0, and ZooKeeper 3.4.10
 Presto: Presto 0.188 with Hadoop 2.8.3 HDFS and Hive 2.3.2 Metastore
 Spark: Spark 2.2.1 on Hadoop 2.8.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3

Use AWS Glue Data Catalog for table metadata

Hardware configuration

Instance type ▼

Number of instances (1 master and 2 core nodes)

Figure 94: Software Configuration

Security and access

EC2 key pair ▼ Learn how to create an

Permissions Default Custom
Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role

EC2 instance profile

Figure 95: Security

If you've selected cluster launch mode, ad-hoc processing of data can now occur on cluster. To get started go to the EMR console page. From here you can then select the 'Clusters' menu on the left side of the screen which will then show you a list of current EMR clusters you have setup: [Figure 96](#). Click on the name of the cluster you would like to run processing steps on and then navigate over to the 'Steps' tab on the following page: see [Figure 97](#). You should see a button named 'Add step' which you can then select to setup the type of step you would like to run for data processing. The step types available to create vary based on the applications that were installed during the creation of the cluster. In this example, only the core Hadoop applications were installed, so the step types are limited to custom JAR, Hive, Pig and streaming programs. Each step type then has a set of parameters that need to be populated before the step can be created. After you've decided on a step type and have populated all of the required step parameters, click the 'Add' button which will then create and run the step on the cluster, see [Figure 98](#). This process can be repeated as needed and there are additional ways to submit up to 256 active steps that can be explored but is beyond the scope of this section [\[41\]](#).

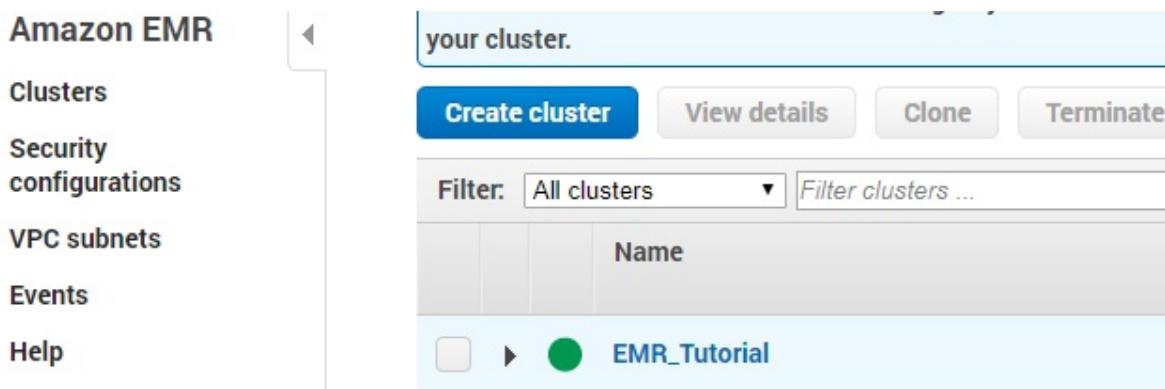


Figure 96: Cluster Create

Cluster: EMR_Tutorial Waiting Cluster ready after last step completed.

The screenshot shows the 'Steps' tab selected in the AWS EMR Cluster interface. A single step is listed:

ID	Name
s-13CWQVQJ070VE	Setup hadoop debugging

Figure 97: Cluster Steps

The screenshot shows the 'Add step' dialog for a 'Hive program'. The form fields are as follows:

- Step type: Hive program
- Name: Hive program
- Script S3 location*: s3://<bucket-name>/<path-to-file>
- Input S3 location: s3://<bucket-name>/<folder>/
- Output S3 location: s3://<bucket-name>/<folder>/
- Arguments: (empty text area)
- Action on failure: Continue

At the bottom right are 'Cancel' and 'Add' buttons.

Figure 98: Additional Steps

12.4.2.3 Teardown

Once completing the steps in the section, it is recommended that you cleanup what you've created in order to avoid high costs of usage. The S3 bucket you created along with the EMR cluster itself will need to be removed. Start with the termination of the cluster by going to main EMR page and selecting the 'Cluster' option on the left. Select the check box for the name of the cluster you wish to terminate and click the 'Terminate' button as the folowing picture shows: [Figure 99](#).

It will then prompt you to confirm this cluster's termination which you will verify and continue. This will place the cluster in a 'Terminating' state and eventually move to a 'Terminated' state. Terminated clusters will remain viewable in the console for two months. You can then proceed to the S3 console page. Before you can delete buckets, you have to delete all of the folders and files contained within that bucket. To do this, click on the bucket name which will then show the subfolders contained within the bucket: [Figure 100](#). Check the box next to all of the subfolder names, select the 'More' button in the menu above and from that menu select 'Delete'. Once all of the folders and files are gone, navigate back to the main S3 page, click the row of the bucket name you wish to delete and select the 'Delete' button. You will then be prompted to enter the name of the bucket you wish to delete and select the 'Confirm' button before the deletion occurs. Once your EMR cluster has been successfully terminated and all buckets created during the section have been deleted, you can then be certain that no additional costs will continue to accrue based on the work performed in this section [\[42\]](#).

Create cluster			View details	Clone	Terminate
Filter:	All clusters	Filter clusters ...	2 clusters (all loaded) C		
	Name		ID	Status	
<input type="checkbox"/>	EMR_Tutorial		j-16Q1RBZ33UWJ	Terminating	User request

Figure 99: Terminate

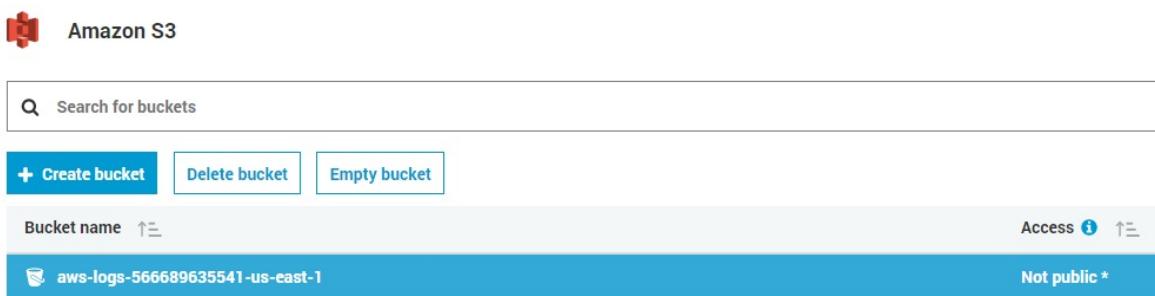


Figure 100: Bucket

12.4.3 TWISTER



○ TODO TA: complete twister documentation

12.4.3.1 Resources

- <http://www.iterativemapreduce.org/>
- <http://www.cs.allegheny.edu/sites/amohan/teaching/CMPSC441/>
- <https://dl.acm.org/citation.cfm?id=1851593>

12.4.4 TWISTER2 INSTALLATION



12.4.4.1 Prerequisites

○ TODO fix the section links

- Operating Systems = [Red Hat Enterprise Linux Server release 7, Ubuntu 14.04 , Ubuntu 16.04] We only use Ubuntu 16.04
- Java (Jdk 1.8) Covered in Section [\[s:hadoop-local-installation\]](#).
- G++ Compiler `sudo apt-get install g++`
- Maven Installation Explained in Section [Maven](#)
- OpenMPI Installation Explained in Section [OpenMPI](#)
- Bazel Build Installation Explained in Section [Bazel](#)
- Additional Libraries Explained in Section [Twister Extra](#)

12.4.4.1.1 Maven Installation

Execute the following commands to install Maven locally.

```
mkdir -p ~/cloudmesh/bin/maven
cd ~/cloudmesh/bin/maven
wget http://mirrors.ibiblio.org/apache/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz
tar xzf apache-maven-3.5.2-bin.tar.gz
```

Adding environmental variables

```
emacs ~/.bashrc
```

Add the following line at the end of the file.

```
MAVEN_HOME=~/cloudmesh/bin/maven/apache-maven-3.5.2
PATH=$MAVEN_HOME/bin:$PATH
export MAVEN_HOME PATH

source ~/.bashrc
```

12.4.4.1.2 OpenMPI Installation

For Twister2, the recommended version is `open MPI 3.0.0`. Execute the following commands to install Open MPI locally.

```
mkdir -p ~/cloudmesh/bin/openmpi
cd ~/cloudmesh/bin/openmpi
wget https://www.open-mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.gz
```

Add environmental variables to bashrc file.

```
emacs ~/.bashrc
mkdir ~/cloudmesh/bin/openmpi/build
BUILD=~/cloudmesh/bin/openmpi/build
OMPI_300=~/cloudmesh/bin/openmpi
PATH=$BUILD/bin:$PATH
LD_LIBRARY_PATH=$BUILD/lib:$LD_LIBRARY_PATH
export BUILD OMPI_300 PATH LD_LIBRARY_PATH

source ~/.bashrc
```

Next build the OpenMPI 3.0.0,

```
cd $OMPI_300
./configure --prefix=$BUILD --enable-mpi-jav
make;make install
```

Make sure the installation is successful by executing,

```
mpirun --version
```

Then you will see an output,

```
mpirun (Open MPI) 3.0.0
```

Install the following command to add this as an maven artifact,

```
mvn install:install-file -DcreateChecksum=true -Dpackaging=jar -Dfile=$OMPI_300/ompi/mpi/
```



12.4.4.1.3 Bazel Installation

For this installation, Bazel 0.8.1 is recommended. Execute the following commands to install Bazel,

```
mkdir -p ~/cloudmesh/bin/bazel
cd ~/cloudmesh/bin/bazel
wget https://github.com/bazelbuild/bazel/releases/download/0.8.1/bazel-0.8.1-installer-linux-x86_64.sh
chmod +x bazel-0.8.1-installer-linux-x86_64.sh
./bazel-0.8.1-installer-linux-x86_64.sh --user
export PATH=$HOME/bin:$PATH
```

12.4.4.1.4 Install Extras

Install the other requirements as follows,

```
sudo apt-get install git build-essential automake cmake libtool-bin zip libunwind-setjmp
sudo apt-get install python-dev python-pip
```

Now you have successfully installed the required packages. Let us compile Twister2.

12.4.4.2 Compiling Twister2

First clone the repository from Github.

```
mkdir ~/cloudmesh/twister2
cd ~/cloudmesh/twister2
git clone git@github.com:DSC-SPIDAL/twister2.git
cd twister2
```

Now compile the code as follows,

```
bazel build --config=ubuntu twister2/...
```

In order to build packages run the following commands,

```
bazel build --config=ubuntu //scripts/package:tarpkgs
```

You can extract the `bazel-bin/scripts/package/twister2-client.tar.gz` to run Twister2.

12.4.5 TWISTER2 EXAMPLES



- TODO TA: complete twister 2 In this section we will be discussing some examples from Twister2 Framework.

12.4.5.1 Introduction to Twister2

12.4.5.2 Twister2 Examples

12.4.6 HARP



- TODO TA: complete harp section

12.4.7 HADOOP RDMA



This section was copied with permission from:
<https://www.chameleoncloud.org/appliances/17/docs/>

The CentOS 7 SR-IOV RDMA-Hadoop appliance is built based on CC-CentOS7 appliance. In this appliance, it also contains a CentOS 7 Virtual Machine image, a VM startup script and a Hadoop cluster launch script, so that users can launch VMs with SR-IOV in order to run RDMA-Hadoop across these VMs on SR-IOV enabled InfiniBand clusters.

- Image name: CC-CentOS7-RDMA-Hadoop
- Default user account: cc
- Remote access: Key-Based SSH
- Root access: passwordless sudo from the cc account
- Chameleon admin access: enabled on the ccadmin account
- Cloud-init enabled on boot: yes
- Repositories (Yum): EPEL, RDO (OpenStack)
- Installed packages:
- Rebuilt kernel to enable IOMMU
- Mellanox SR-IOV drivers for InfiniBand
- KVM hypervisor
- Standard development tools such as make, gcc, gfortran, etc.

- Config management tools: Puppet, Ansible, Salt
- OpenStack command-line clients
- Included VM image name: chameleon-rdma-hadoop-appliance.qcow2
- Included VM startup script: start-vm.sh
- Included Hadoop cluster launch script: launch-hadoop-cluster.sh
- Default VM root password: nowlab

Please refer to the bare metal user guide for documentation on how to reserve and provision resources using the appliance of CC-CentOS7-RDMA-Hadoop.

12.4.7.1 Launching a Virtual Hadoop Cluster on Bare-metal InfiniBand Nodes with SR-IOV on Chameleon

We provide a CentOS 7 VM image (chameleon-rdma-hadoop-appliance.qcow2) and a Hadoop cluster launch script (launch-hadoop-cluster.sh) to facilitate users to setup Virtual Hadoop Clusters effortlessly.

First, launch bare-metal nodes using the RDMA-Hadoop Appliance and select one of the nodes as the bootstrap node. This node will serve as the host for the master node of the Hadoop cluster and will also be used to setup the entire cluster. Now, ssh to this node. Before you can launch the cluster, you have to download your OpenStack credentials file (see how to download your credentials file). Then, create a file (henceforth referred to as ips-file) with the ip addresses of the bare-metal nodes you want to launch your Hadoop cluster on (excluding the bootstrap node), each on a new line. Next, run these commands as root:

```
[root@host]$ cd /home/cc
[root@host]$ ./launch-hadoop-cluster.sh <num-of-vms-per-node> <num-of-MB-per-VM> <num-of-
```

The launch cluster script will launch VMs for you, then install and configure Hadoop on these VMs. Note that when you launch the

cluster for the first time, a lot of initialization is required. Depending on the size of your cluster, it may take some time to setup the cluster. After the cluster setup is complete, the script will print an output telling you that the cluster is setup and how you can connect to the Hadoop master node. Note that the minimum required memory for each VM is 8,192 MB. The Hadoop cluster will already be setup for use. For more details on how to use the RDMA-Hadoop package to run jobs, please refer to its user guide.

12.4.7.2 Launching Virtual Machines Manually

We provide a CentOS 7 VM image (chameleon-rdma-hadoop-appliance.qcow2) and a VM startup script (start-vm.sh) to facilitate users to launch VMs manually. Before you can launch a VM, you have to create a network port. To do this, source your OpenStack credentials file (see how to download your credentials file) and run this command:

```
[user@host]$ neutron port-create sharednet1
```

Note the MAC address and IP address are in the output of this command. You should use this MAC address while launching a VM and the IP address to ssh to the VM. You also need the PCI device ID of the virtual function that you want to assign to the VM. This can be obtained by running "lspci | grep Mellanox" and looking for the device ID (with format - XX:XX.X) of one of the virtual functions as shown below:

```
[cc@host]$ lspci | grep Mellanox
03:00.0 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3]
03:00.1 Network controller: Mellanox Technologies MT27500/MT27520 Family [ConnectX-3/Connec
...
1
```

The PCI device ID of the Virtual Function is 03:00:1 in the above example.

Now, you can launch a VM on your instance with SR-IOV using the provided VM startup script and corresponding arguments as follows with the root account.

```
[root@host]$ ./start-vm.sh <vm-mac> <vm-ifname> <virtual-function-device-id>
```

Please note that and are the ones you get from the outputs of above commands. And is the name of VM virtual NIC interface. For example:

```
[root@host]$ ./start-vm.sh fa:16:3e:47:48:00 tap0 03:00:1
```

You can also edit corresponding fields in VM startup script to change the number of cores, memory size, etc.

You should now have a VM running on your bare metal instance. If you want to run more VMs on your instance, you will have to create more network ports. You will also have to change the name of VM virtual NIC interface to different ones (like tap1, tap2, etc.) and select different device IDs of virtual functions.

12.4.7.3 Extra Initialization when Launching Virtual Machines

In order to run RDMA-Hadoop across VMs with SR-IOV, and keep the size of VM image small, extra initialization will be executed when launching VM automatically, which includes:

- Detect Mellanox SR-IOV drivers, download and install it if nonexistent
 - * Detect Java package installed, download and install if non-existent
 - * Detect RDMA-Hadoop package installed, download and install if non-existent

After finishing the extra initialization procedure, you should be able to run Hadoop jobs with SR-IOV support across VMs. Note that this initialization will be done automatically. For more details about the RDMA-Hadoop package, please refer to its user guide.

12.4.7.4 Important Note for Tearing Down Virtual Machines and Deleting Network Ports

Once you are done with your experiments, you should kill all the launched VMs and delete the created network ports. If you used the launch-hadoop-cluster.sh script to launch VMs, you can do this by

running the kill-vms.sh script as shown below. This script will kill all launched VMs and also delete all the created network ports.

```
[root@host]$ cd /home/cc  
[root@host]$ ./kill-vms.sh <ips-file> <openstack-credentials-file>  
\end{verbatim}
```

If you launched VMs using the start-vm.sh script, you should first manually kill all the VMs.

```
[user@host]$ neutron port-delete PORT
```

Please note that it is important to delete unused ports after experiments.

13 CONTAINERS



13.1 INTRODUCTION TO CONTAINERS



🎓 Learning Objectives

- Knowing what a container is.
- Differentiating Containers from Virtual Machines.
- Understanding the historical aspects that lead to containers.

This section covers an introduction to containers that is split up into four parts. We discuss microservices, serverless computing, Docker, and kubernetes.

13.1.1 Motivation - Microservices

We discuss the motivation for containers and contrast them to virtual machines. Additionally we provide a motivation for containers as they can be used to microservices.

[Container 11:01 Container A](#)

13.1.2 Motivation - Serverless Computing

We enhance our motivation while contrasting containers and microservices while relating them to serverless computing. We anticipate that serverless computing will increase in importance over the next years

[Container 15:08 Container B](#)

13.1.3 Docker

In order for us to use containers, we go beyond the historical

motivation that was introduced in a previous section and focus on Docker a predominant technology for containers on Windows, Linux, and macOS

🎬 [Container 40:09 Container C](#)

13.1.4 Docker and Kubernetes

We continue our discussion about docker and introduce kubernetes, allowing us to run multiple containers on multiple servers building a cluster of containers.

🎬 [Container 50:14 Container D](#)

13.1.5 Resources

- Container Orchestration Tools: Compare Kubernetes vs Docker Swarm <https://platform9.com/blog/compare-kubernetes-vs-docker-swarm/>
- Gentle introduction to Containers <https://www.slideshare.net/jpetazzo/introduction-docker-linux-containers-lxc>

13.1.5.1 Tutorialspoint

Several tutorials on docker that can help you understand the concepts in more detail

- <https://www.tutorialspoint.com/docker/index.htm>
- https://www.tutorialspoint.com/docker/docker_tutorial.pdf
- https://www.tutorialspoint.com/docker/docker_pdf_version.htm

13.2 DOCKER



13.2.1 INTRODUCTION TO DOCKER



Docker is the company driving the container movement and the only

container platform provider to address every application across the hybrid cloud. Today's businesses are under pressure to digitally transform but are constrained by existing applications and infrastructure while rationalizing an increasingly diverse portfolio of clouds, datacenters and application architectures. Docker enables true independence between applications and infrastructure and developers and IT ops to unlock their potential and creates a model for better collaboration and innovation. An overview of docker is provided at

- <https://docs.docker.com/engine/docker-overview/>

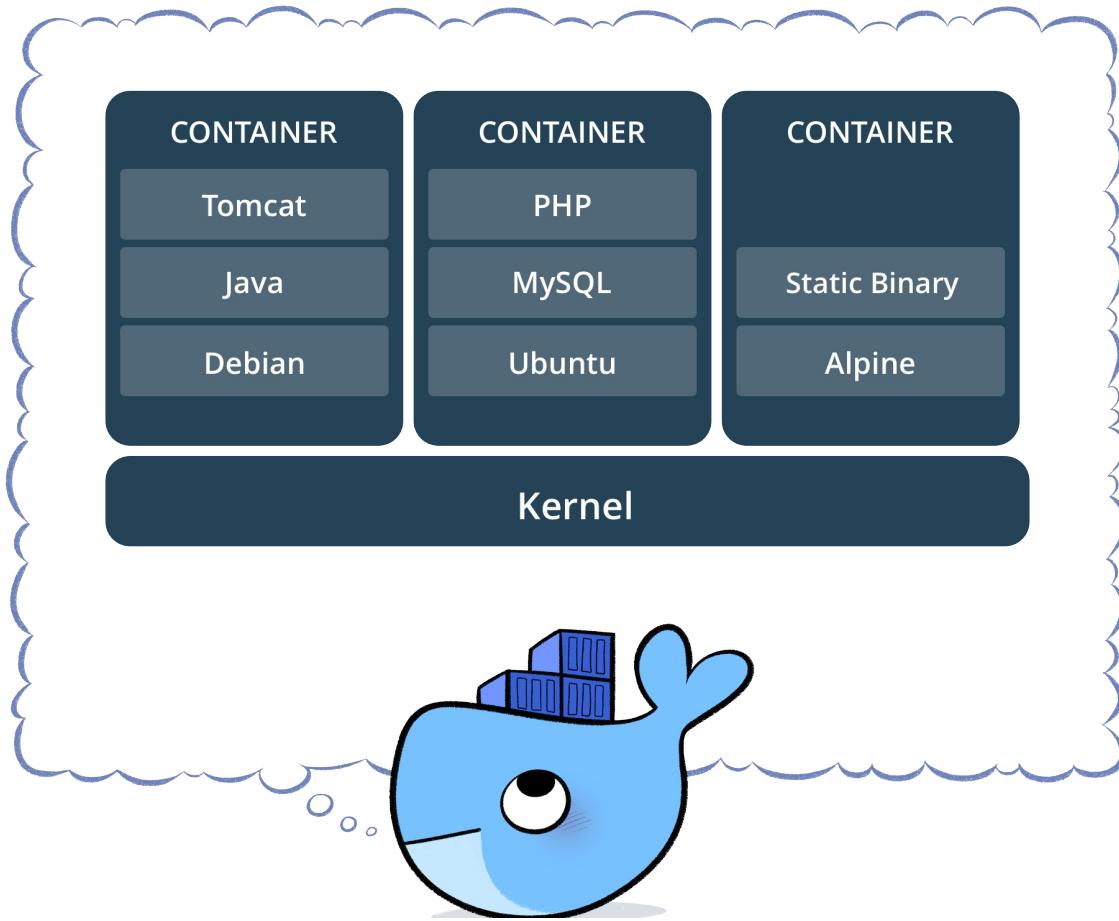


Figure 101: Docker Containers [\[Image Source\]](#)

[Figure 101](#) shows how docker containers fit into the system ## Docker platform

Docker provides users and developers with the tools and technologies that are needed to manage their application development using containers. Developers can easily setup different environments for development, testing and production.

13.2.1.1 Docker Engine

The Docker engine can be thought of as the core of the docker runtime. The docker engine mainly provides 3 services. [Figure 102](#) shows how the docker engine is composed.

- A long running server which manages the containers

- A REST API
- A command line interface

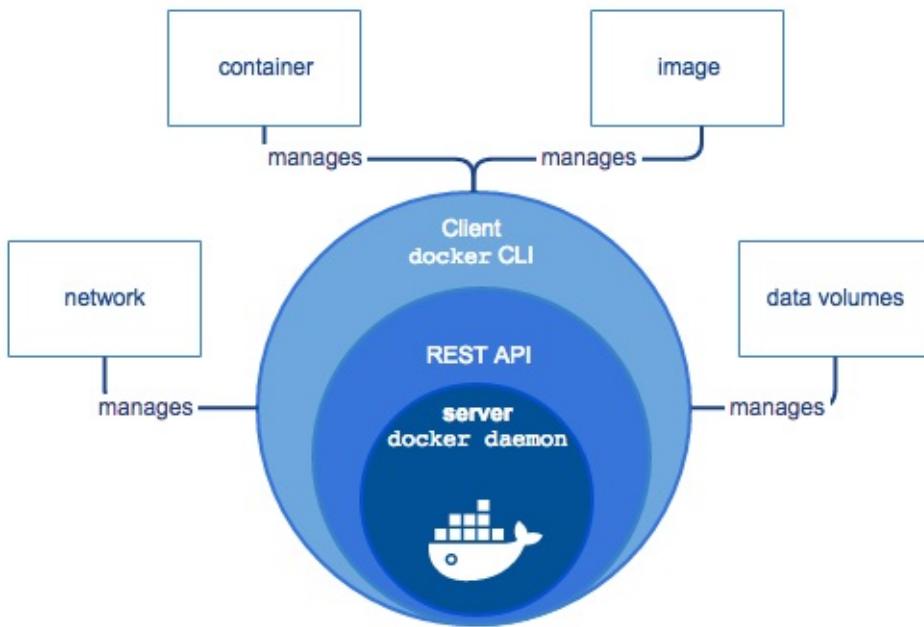


Figure 102: Docker Engine Component Flow [\[Image Source\]](#)

13.2.1.2 Docker Architecture

The main concept of the docker architecture is based on the simple client-server model. Docker clients communicate with the Docker server also known as the Docker daemon to request various resources and services. The daemon manages all the background tasks that need to be performed to complete client requests. Managing and distributing containers, running the containers, building containers, etc. are responsibilities of the Docker daemon. [Figure 103](#) shows how the docker architecture is setup. The client module and server can run either in the same machine or in separate machines. In the latter case the communication between the client and server are done through the network.

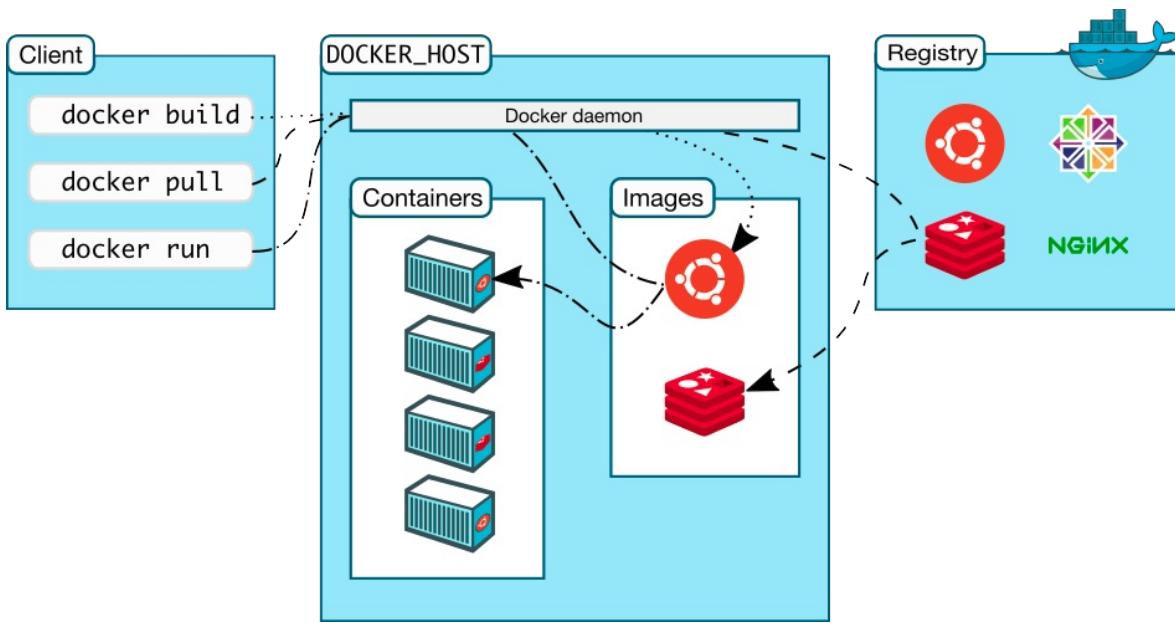


Figure 103: Docker Architecture [\[Image Source\]](#)

13.2.1.3 Docker Survey

In 2016 Docker Inc. surveyed over 500 Docker developers and operations experts in various phases of deploying container-based technologies. The result is available in the The Docker Survey 2016 as seen in [Figure 104](#).

- <https://www.docker.com/survey-2016>

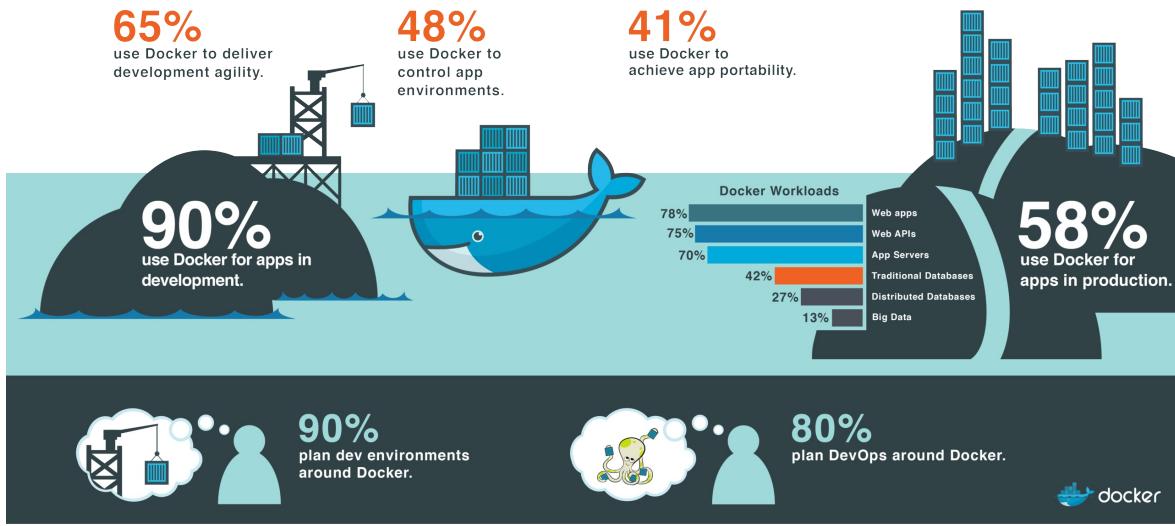


Figure 104: Docker Survey Results 2016 [\[Image Source\]](#)

13.2.2 RUNNING DOCKER LOCALLY



The official installation documentation for docker can be found by visiting the following Web page:

- <https://www.docker.com/community-edition>

Here you will find a variety of packages, one of which will hopefully suitable for your operating system. The supported operating systems currently include:

- OSX, Windows, Centos, Debian, Fedora, Ubuntu, AWS, Azure

Please chose the one most suitable for you. For your convenience we provide you with installation instructions for OSX (Section [Docker on OSX](#)), Windows 10 (Section [Docker on Windows](#)) and Ubuntu (Section [Docker on ubuntu](#)).

13.2.2.1 Instillation for OSX

The docker community edition for OSX can be found at the following link

- <https://store.docker.com/editions/community/docker-ce->

[desktop-mac](#)

We recommend that at this time you get the version Docker CE for MAC (stable)

- <https://download.docker.com/mac/stable/Docker.dmg>

Clicking on the link will download a dmg file to your machine, that you than will need to install by double clicking and allowing access to the dmg file. Upon installation a whale in the top status bar shows that Docker is running, and you can access it via a terminal.



Docker integrated in the menu bar on OSX

13.2.2.2 Installation for Ubuntu

In order to install Docker community edition for Ubuntu, you first have to register the repository from where you can download it. This can be achieved as follows:

```
local$ sudo apt-get update
local$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
local$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
local$ sudo apt-key fingerprint 0EBFCD88
local$ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    local$(lsb_release -cs) \
    stable"
```

Now that you have configured the repository location, you can install it after you have updated the operating system. The update and install is done as follows:

```
local$ sudo apt-get update
local$ sudo apt-get install docker-ce
local$ sudo apt-get update
```

Once installed execute the following command to make sure the

installation is done properly

```
local$ sudo systemctl status docker
```

This should give you an output similar to below.

```
docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2018-10-03 13:02:04 EDT; 15min ago
     Docs: https://docs.docker.com
 Main PID: 6663 (dockerd)
    Tasks: 39
```

13.2.2.3 Installation for Windows 10

Docker needs Microsoft's Hyper-V to be enabled, but it will impact running the virtual machines

Steps to Install

- Download Docker for Windows(Community Edition) from the following [link](https://download.docker.com/win/stable/Docker%20for%20Windows)
[https://download.docker.com/win/stable/Docker%20for%20W](https://download.docker.com/win/stable/Docker%20for%20Windows)
- Follow the Wizard steps in the installer
- Launch docker
- Docker usually launches automatically during windows startup.

13.2.2.4 Testing the Install

To test if it works execute the following commands in a terminal:

```
local$ docker version
```

You should see an output similar to

```
docker version

Client:
 Version:      17.03.1-ce
 API version:  1.27
 Go version:   go1.7.5
 Git commit:   c6d412e
 Built:        Tue Mar 28 00:40:02 2017
 OS/Arch:      darwin/amd64
```

```
Server:  
Version: 17.03.1-ce  
API version: 1.27 (minimum version 1.12)  
Go version: go1.7.5  
Git commit: c6d412e  
Built: Fri Mar 24 00:00:50 2017  
OS/Arch: linux/amd64  
Experimental: true
```

To see if you can run a container use

```
local$ docker run hello-world
```

Once executed you should see an output similar to

```
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
78445dd45222: Pull complete  
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a .....
```

Status: Downloaded newer image for
hello-world:latest

```
Hello from Docker!  
This message shows that your installation appears  
to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "[hello-world](#)" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
local$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a [free](#) Docker ID:

<https://cloud.docker.com/>

For more examples and ideas, visit:
<https://docs.docker.com/engine/userguide/>

13.2.3 DOCKERFILE



In order for us to build containers, we need to know what is in the container and how to create an image representing a container. To do this a convenient specification format called `Dockerfile` can be used. Once a `Dockerfile` is created, we can build images from it

We showcase here the use of a dockerfile on a simple example using a REST service.

This example is copied from the official docker documentation hosted at

- <https://docs.docker.com/get-started/part2/#publish-the-image>

13.2.3.1 Specification

It os best to start with an empty directory in which we create a Dockerfile.

```
local$ mkdir ~/cloudmesh/docker  
local$ cd ~/cloudmesh/docker
```

Next, we create an empty file called `Dockerfile`

```
local$ touch Dockerfile
```

We copy the following contents into the Dockerfile and after that create a simple REST service

```
# Use an official Python runtime as a parent image  
FROM python:3.7-slim  
  
# Set the working directory to /app  
WORKDIR /app  
  
# Copy the current directory contents into the container at /app  
COPY . /app  
  
# Install any needed packages specified in requirements.txt  
RUN pip install --trusted-host pypi.python.org -r requirements.txt  
  
# Make port 80 available  
EXPOSE 80  
  
# Run app.py when the container launches
```

```
CMD ["python", "app.py"]
```

We also create a `requirements.txt` file that we need for installing the necessary python packages

```
Flask
```

The example application we use here is a student info served via a RESTful service implemented using python flask. It is stored in the file `app.py`

```
from flask import Flask, jsonify
import os

app = Flask(__name__)

@app.route('/student/albert')
def alberts_information():
    data = {
        'firstname': 'Albert',
        'lastname': 'Zweistsein',
        'university': 'Indiana University',
        'email': 'albert@example.com'
    }
    return jsonify(**data)

if __name__ == '__main__':
    app.run(host="0.0.0.0", port=80)
```

To build the container, we can use the following command:

```
local$ docker build -t students .
```

To run the service open a new window and cd into the directory where you code is located. Now say

```
local$ docker run -d -p 4000:80 students
```

Your docker container will run and you can visit it by using the command

```
local$ curl http://localhost:4000/student/albert
```

To stop the container do a

```
local$ docker ps
```

and locate the id of the container, e.g., 2a19776ab812, and then run this

```
local$ docker stop 2a19776ab812
```

To delete the docker container image, you must first stop all instances using it and then remove the image. You can see the images with the command

```
local$ docker images
```

Then you can locate all containers using that image while looking in the IMAGE column or using a simple fgrep in case you have many images. Stop the containers using that image and then you can say

```
local$ docker rm 74b9b994c9bd
```

while the number is the container id

Once you killed all containers using that image, you can remove the image with the `rmi` command.

```
local$ docker rmi 8b3246425402
```

13.2.3.2 References

The reference documentation about docker files can be found at

- <https://docs.docker.com/engine/reference/builder/>

13.2.4 Docker Hub



Docker Hub is a cloud-based registry service which provides a “centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline” [43]. There are both private and public repositories. Private repository can only be used by people within their own organization.

Docker Hub is integrated into Docker as the default registry. This

means that the docker pull command will initialize the download automatically from Docker Hub [44]. It allows users to download (pull), build, test and store their images for easy deployment on any host they may have [43].

13.2.4.1 Create Docker ID and Log In

A log-in is not necessary for pulling Docker images from the Hub but it is necessary for pushing images to dockerhub for sharing. Thus to store images on Docker hub you need to create an account by visiting [Docker Hub Web page](#). Dockerhub offers in general a free account, but it has restrictions. The free account allows you to share images that you distribute publically, but it only allows one private Docker Hub Repository. In case you need more, you will need to upgrade to a paid plan.

For the rest of the tutorial we assume that you use the environment variable DOCKERHUB to indicate yourusername. It is easiest if you set it in your shell with

```
local$ export DOCKERHUB=<PUT YOUR DOCKER USERNAME HERE>
```

13.2.4.2 Searching for Docker Images

There are two ways to search for Docker images on Docker Hub:

One way is to use the Docker command line tool. We can open a terminal and run the docker search command. For example, the following command searches for centOS images:

```
local$ sudo docker search centos
```

you will see output similar to:

NAME	DESCRIPTION	STAR	OFFICIAL	AUTOMATED
centos	Official CentOS	4130	[OK]	

ansible/centos7	Ansible on Centos7	105	[OK]
-----------------	-----------------------	-----	------

...

If you do not want to use sudo with docker command each time you need to add the current user into the docker group. You can do that using the following command.

```
local$ sudo usermod -aG docker ${USER}  
local$ su - ${USER}
```

This will prompt you to enter the password for the current user. Now you should be able to execute the above command without using sudo.

Official repositories in dockerhub are public, certified repositories from vendors and contributors to Docker. They contain Docker images from vendors like Canonical, Oracle, and Red Hat that you can use as the basis to build your applications and services. There is one official repository in this list, the first one, centos.

The other way is to search via the Web Search Box at the top of the Docker web page by typing the keyword. The search results can be sorted by number of stars, number of pulls, and whether it is an official image. Then for each search result, you can verify the information of the image by clicking the details button to make sure this is the right image that fits your needs.

13.2.4.3 Pulling Images

A particular image (take centos as an example) can be pulled using the following command:

```
local$ docker pull centos
```

Tags can be used to specify the image to pull. By default the tag is latest, therefore the previous command is the same as the following:

```
local$ docker pull centos:latest
```

You can use a different tag:

```
local$ docker pull centos:6
```

To check the existing local docker images, run the following command:

```
local$ docker images
```

The results show:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
centos	latest	26cb1244b171	2 weeks ago	195MB
centos	6	2d194b392dd1	2 weeks ago	195MB

13.2.4.4 Create Repositories

In order to push images to Docker Hub, you need to have a account and create a repository.

When you first create a Docker Hub user, you see a Get started with Docker Hub screen, from which you can click directly into Create Repository. You can also use the Create menu to Create Repository. When creating a new repository, you can choose to put it in your Docker ID namespace, or that of any organization that you are in the owners team [45].

As an example, we created a repository `cloudtechnology` with the namespace `$DOCKERHUB` (here `DOCKERHUB` is your docker hub username). Hence the full name is `$DOCKERHUB/cloudtechnology`

13.2.4.5 Pushing Images

To push an image to the repository created, the following steps can be followed.

First, log into Docker Hub from the command line by specifying the username. If you encounter permission issues please use `sudo` in front of the command

```
$ docker login --username=$DOCKERHUB
```

Enter the password when prompted. If everything worked you will get a message similar to:

```
Login Succeeded
```

Second, check the image ID using:

```
$ docker images
```

the result looks similar to:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
cloudmesh-nlp	latest	1f26a5f7a1b4	10 days ago	1.79GB
centos	latest	26cb1244b171	2 weeks ago	195MB
centos	latest	2d194b392dd1	2 weeks ago	195MB

Here, the the image with ID 1f26a5f7a1b4 is the one to push to Docker Hub. You can choose another image instead if you like.

Third, tag the image

```
$ docker tag 1f26a5f7a1b4 $DOCKERHUB/cloudmesh:v1.0
```

Here we have used a version number as a tag. However another good way of adding a tag is to use a keyword/tag that will help you understand what this container should be used in conjunction with, or what it represents.

Fourth, now the list of images will look something like

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
cloudmesh-nlp	latest	1f26a5f7a1b4	10 d ago	1.79

\$DOCKERHUB/cloudmesh	v1.0	1f26a5f7a1b4	10 d ago	1.79
centos	latest	26cb1244b171	2 w ago	195
centos	latest	2d194b392dd1	2 w ago	195

Fifth, Now you can see an images under the name `$DOCKERHUB/cloudmesh`, we now need to push this image to the repository that we created on the docker hub website. For that execute the following command.

```
$ docker push $DOCKERHUB/cloudmesh
```

It shows something similar to, to make sure you can check on docker hub if the images that was pushed is listed in the repository that we created.

```
The push refers to repository [docker.io/$DOCKERHUB/cloudmesh]
18f9479cf2c: Pushed
e9ddee98220b: Pushed
...
db584c622b50: Mounted from library/ubuntu
a94e0d5a7c40: Mounted from library/ubuntu
...
v1.0: digest: sha256:305b0f911077d9d6aab4b447b... size: 3463
```

Sixth, now the image is available on Docker Hub. Everyone can pull it since it is a public repository by using command:

```
$ docker pull USERNAME/cloudmesh
```

Please remember that the `USERNAME` is the username for the user that makes this image publically available. If you are the user you will see the value being the one from `$DOCKERHUB`, If not you will see here the username of the user uploading the image

13.2.4.6 Resources

- The offical [Overview of Docker Hub \[43\]](#)
- Information about using docker repositories can be found at [Repositories on Docker Hub \[45\]](#)
- [How to Use DockerHub \[44\]](#)
- [Docker Tutorial Series \[46\]](#)

13.3 DOCKER CLUSTERS



In this section we present mechanisms for managing containers across multiple hosts. This includes docker swarm and kubernetes.

13.3.1 DOCKER SWARM



A swarm is a group of machines that are running Docker and are joined into a cluster. Docker commands are executed on a cluster by a swarm manager. The machines in a swarm can be physical or virtual. After joining a swarm, they are referred to as nodes.

13.3.1.1 Terminology

In this section if a command is prefixed with `local$` it means the command is to be executed on your local machine. If it is prefixed with either `master` or `worker` that means the command is to be executed from within a virtual machine that was created.

13.3.1.2 Creating a Docker Swarm Cluster

A swarm is made up of multiple nodes, which can be either physical or virtual machines. We use `master` as the name of the host that is run as master and `worker-1` as a host run as a worker, where the number indicates the i-th worker. The basic steps are:

1. run

```
master$ docker swarm init
```

to enable swarm mode and make your current machine a swarm manager,

2. then run

```
worker-1$ docker swarm join
```

on other machines to have them join the swarm as workers.

Choose a tab below to see how this plays out in various contexts. We use VMs to quickly create a two-machine cluster and turn it into a swarm.

13.3.1.3 Create a Swarm Cluster with VirtualBox

In case you do not have access to multiple physical machines, you can create a virtual cluster on your machine with the help of virtual box. Instead of using `vagrant` we can use the built in `docker-machine` command to start several virtual machines.

If you do not have `virtualbox` installed on your machine install it on your machine. Additionally you would require `docker-machine` to be installed on your local machine. To install `docker-machine` on please follow instructions at the docker documentation at [Install Docker Machine](#)

To create the virtual machines you can use the command as follows:

```
local$ docker-machine create --driver virtualbox master  
local$ docker-machine create --driver virtualbox worker-1
```

To list the VMs and get their ip addresses. Use this command to list the machines and get their IP addresses.

```
local$ docker-machine ls
```

13.3.1.4 Initialize the Swarm Manager Node and Add Worker Nodes

The first machine acts as the manager, which executes management commands and authenticates workers to join the swarm, and the second is a worker.

To instruct the first vm to become the master, first we need to login to the vm that was named `master`. To login you can use `ssh`, execute the following command on your local machine to login to the `master` vm.

```
local$ docker-machine ssh master
```

Now since we are inside the `master` vm we can configure this vm as the docker swarm manager. Execute the following command within the `master` vm in initialize swarm

```
master$ docker swarm init
```

If you get an error stating something similar to “could not choose an IP address to advertise since this system has multiple addresses on different interfaces”, use the following command instead. To find the IP address execute the command `ifconfig` and pick the ip address which is most simmilar to `192.x.x.x`.

```
master$ docker swarm init --advertise-addr 192.x.x.x
```

The output wil look like this, where `IP-myvm1` is the ip address of the first vm

```
master$ Swarm initialized: current node (p6hmohoeuggtwqj8xz91zbs5t) is now  
a manager.
```

To add a worker to this swarm, run the following command:

```
worker-1$ docker swarm join --token SWMTKN-1-5c3anju1pxw94054r3vx0v7j4obyuggfu2cmesnx  
192.168.99.100:2377
```

To add a manager to this swarm, run '`docker swarm join-token manager`' and follow the instru

Now that we have the docker swarm manager up we can add worker machines to the swarm. The command that is printed in the output shown above can be used to join workers to the manager. Please note that you need to use the output command that is generated when you run `docker swarm init` since the token values will be different.

Now we need to use a separate shell to login to the worker vm that we created. Open up a new shell (or terminal) and use the following command to ssh into the `worker`

```
local$ docker-machine ssh worker-1
```

Once you are in the `worker` execute the following command to join `worker` to the swam manager.

```
worker-1$ docker swarm join --token  
SWMTKN-1-5c3anju1pwx94054r3vx0v7j4obyuggfu2cmesnx 192.168.99.100:2377
```

The generic version of the command would be as follows, you need to fill in the correct values to values marked as '<>' to execute the command.

```
worker-1$ docker swarm join --token <token> <myvm ip>:<port>
```

You will see an output stating that this machine joined the docker swarm.

```
This node joined a swarm as a worker.
```

If you want to add another node as a manager to the current swarm you can execute the following command and follow the instructions. However this is not needed for this exercise.

```
newvm$ docker swarm join-token manager
```

Run `docker-machine ls` to verify that `worker` is now the active machine, as indicated by the asterisk next to it.

```
local$ docker-machine ls
```

If the astrix is not present execute the following command

```
local$ sudo sh -c 'eval "$(docker-machine env worker-1)"; docker-machine ls'
```

The output will look similar to

NAME	ACTIVE	DRIVER	STATE	URL	SWARM	DOCKER
master	-	virtualbox	Running	tcp://192.168.99.100:2376		v18.06.1-ce
worker-1	*	virtualbox	Running	tcp://192.168.99.102:2376		v18.06.1-ce

13.3.1.5 Deploy the application on the swarm manager

Now we can try to deploy a test application. First we need to create a docker configuration file which we will name `docker-compose.yml`. Since we are in the vm we need to create the file using the terminal. follow the steps given below the create and save the file. First log into the `master`

```
local$ docker-machine ssh worker-1
```

Then,

```
master$ vi docker-compose.yml
```

This command will open an editor. Press the `Insert` button to enable editing and then copy paste the following into the document.

```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "4000:80"
    networks:
      - webnet
networks:
  webnet:
```

Then press the `ECS` button and enter `:wq` to save and close the editor.

Once we have the file we can deploy the test application using the following command. which will be executed in the `master`

```
master$ docker stack deploy -c docker-compose.yml getstartedlab
```

To verify the services and associated containers have been distributed between both `master` and `worker`, execute the following command.

```
master$ docker stack ps getstartedlab
```

The output will look similar to

```
```bash
ID NAME IMAGE NODE DESIRED STATE CURRENT STATE
ERROR PORTS wpqtkv69qbee getstartedlab_web.1
username/repo:tag worker-1 Running Preparing 4 seconds ago
whkiecyenuv0 getstartedlab_web.2 username/repo:tag master
```

```
Running Preparing 4 seconds ago 13obecvxohh1 getstartedlab_web.3
username/repo:tag worker-1 Running Preparing 5 seconds ago
76srj0nflagi getstartedlab_web.4 username/repo:tag worker-1
Running Preparing 5 seconds ago ymqoonad5c1f
getstartedlab_web.5 username/repo:tag master Running Preparing 5
seconds ago
```

### 13.3.2 DOCKER AND DOCKER SWARM ON FUTURESYSTEMS



This section is for IU students only that take classes with us.

This section introduces how to run Docker container on FutureSystems. Currently we have deployed Docker swarm on Echo.

#### 13.3.2.1 Getting Access

You will need an account on FutureSystems and be enrolled in an active project. To verify, try to see if you can log into `victor.futuresystems.org`. You need to be a member of a valid FutureSystems project, and had submitted an ssh public key via the FutureSystems portal.

For Fall 2018 classes at IU you need to be in the following project:

<https://portal.futuresystems.org/project/553>

If your access to the victor host has been verified, try to login to the docker swarm head node. To conveniently do this let us define some Linux environment variables to simplify the access and the material presented here. You can place them even in your `.bashrc` or `.bash_profile` so the information gets populated whenever you start a new terminal. If you directly edit the files make sure to execute the `source` command to refresh the environment variables for the current session using `source .bashrc` or `source .bash_profile`. Or you can close the current shell and reopen a new one.

```
local$ export ECHO=149.165.150.76
local$ export FS_USER=<put your futersystem account name here>
```

Now you can use the two variables that were set to login to the Echo server, using the following command

```
local$ ssh $FS_USER@$ECHO
```

**Note: If you have access to india but not the docker swarm system, your project may not have been authorized to access the docker swarm cluster. Send a ticket to FutureSystems ticket system to request this.**

Once logged in to the docker swarm head node, try to run:

```
echo$ docker run hello-world
```

to verify `docker run` works.

### 13.3.2.2 Creating a service and deploy to the swarm cluster

While `docker run` can start a container and you may even attach to its console, the recommended way to use a docker swarm cluster is to create a service and have it run on the swarm cluster. The service will be scheduled to one or many number of the nodes of the swarm cluster, based on the configuration. It is also easy to scale up the service when more swarm nodes are available. Docker swarm really makes it easier for service/application developers to focus on the functionality development but not worrying about how and where to bind the service to some resources/server. The deployment, access, and scaling up/down when necessary, are all managed transparently. Thus achieving the new paradigm of serverless computing.

As an example, the following command creates a service and deploy it to the swarm cluster, if the port is in use the port `9001` used in the command can be changed to an available port.

```
echo$ docker service create --name notebook_test -p 9001:8888 \
 jupyter/datascience-notebook start-notebook.sh
 --NotebookApp.password=NOTEBOOK_PASS_HASH
```

The `NOTEBOOK_PASS_HASH` can be generated in python:

```
>>> import IPython
>>> IPython.lib.passwd("YOUR_SELECTED_PASSWORD")
'sha1:52679cadb4c9:6762e266af44f86f3d170ca1.....'
```

So pass through the string starting with 'sha1:.....'.

The command pulls a published image from docker cloud, starts a container and runs a script to start the service inside the container with necessary parameters. The option "-p 9001:8888" maps the service port inside the container (8888) to an external port of the cluster node (9001) so the service could be accessed from the Internet. In this example, you can then visit the URL:

```
local$ open http://$ECHO:9001
```

to access the Jupyter notebook. Using the specified password when you create the service to login.

Please note the service will be dynamically deployed to a container instance, which would be allocated to a swarm node based on the allocation policy. Docker makes this process transparent to the user and even created mesh routing so you can access the service using the IP address of the management head node of the swarm cluster, no matter which actual physical node the service was deployed to.

This also implies that the external port number used has to be free at the time when the service was created.

Some useful related commands:

```
echo$ docker service ls
```

lists the currently running services.

```
echo$ docker service ps notebook_test
```

lists the detailed info of the container where the service is running.

```
echo$ docker node ps NODE
```

lists all the running containers of a node.

```
echo$ docker node ls
```

lists all the nodes in the swarm cluster.

To stop the service and the container:

```
echo$ docker service rm noteboot_test
```

### 13.3.2.3 Create your own service

You can create your own service and run it. To do so, start from a base image, e.g., a ubuntu image from the docker cloud. Then you could:

- Run a container from the image and attach to its console to develop the service, and create a new image from the changed instance using command 'docker commit'.
- Create a dockerfile, which has the step by step building process of the service, and then build an image from it.

In reality, the first approach is probably useful when you are in the phase of develop and debug your application/service. Once you have the step by step instructions developed the latter approach is the recommended way.

Publish the image to the docker cloud by following this documentation:

- <https://docs.docker.com/docker-cloud/builds/push-images/>

Please make sure no sensitive information is included in the image to be published. Alternatively you could publish the image internally to the swarm cluster.

### 13.3.2.4 Publish an image privately within the swarm cluster

Once the image is published and available to the swarm cluster, you could start a new service from the image similar to the Jupyter

Notebook example.

### 13.3.2.5 Exercises

E.Docker.Futuresystems.1:

Obtain an account on future systems.

E.Docker.Futuresystems.2:

Create a REST service with swagger codegen and run it on the echo cloud (see example in [this section](#) )

## 13.3.3 HADOOP WITH DOCKER



In this section we will explore the Map/Reduce framework using Hadoop provided through a Docker container.

We will showcase the functionality on a small example that calculates minimum, maximum, average and standard deviation values using several input files which contain float numbers.

This section is based on the hadoop release 3.0.3 which includes significant enhancements over the previous version of Hadoop 2.x. Changes include the use of the following software:

- CentOS 7
- systemctl
- Java SE Development Kit 8

A Dockerfile to create the hadoop deployment is available at

\*<https://github.com/cloudmesh-community/book/blob/master/examples/docker/hadoop/3.0.3/Dockerfile>

### 13.3.3.1 Building Hadoop using Docker

You can build hadoop from the Dockerfile as follows:

```
$ mkdir cloudmesh-community
$ cd cloudmesh-community
$ git clone https://github.com/cloudmesh-community/book.git
$ cd book/examples/docker/hadoop/3.0.3
$ docker build -t cloudmesh/hadoop:3.0.3 .
```

The complete docker image for Hadoop consumes 1.5GB.

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
cloudmesh/hadoop	3.0.3	ba2c51f94348	1 hour ago	1.52GB

To use the image interactively you can start the container as follows:

```
$ docker run -it cloudmesh/hadoop:3.0.3 /etc/bootstrap.sh -bash
```

It may take a few minutes at first to download image.

### 13.3.3.2 Hadoop Configuration Files

The configuration files are included in the `conf` folder

### 13.3.3.3 Virtual Memory Limit

IN case you need more memory, you can increase it by changing the parameters in the file `mapred-site.xml`, for example:

- `mapreduce.map.memory.mba` to 4096
- `mapreduce.reduce.memory.mb` to 8192

### 13.3.3.4 hdfs Safemode leave command

A Safemode for HDFS is a read-only mode for the HDFS cluster, where it does not allow any modifications of files and blocks. Namenode disables safe mode automatically after starting up normally. If required, HDFS could be forced to leave the safe mode explicitly by this command:

```
$ hdfs dfsadmin -safemode leave
```

### 13.3.3.5 Examples

We included a statistics and a PageRank examples into the container. The examples are also available in github at

- <https://github.com/cloudmesh-community/book/tree/master/examples/docker/hadoop/3.0.3>

We explain the examples next

#### 13.3.3.5.1 Statistical Example with Hadoop

After we launch the container and use the interactive shell, we can run the statistics Hadoop application which calculates the minimum, maximum, average, and standard derivation from values stored in a number of input files. Figure [Figure 105](#) shows the computing phases in a MapReduce job.

To achieve this, this Hadoop program reads multiple files from HDFS and provides calculated values. We walk through every step from compiling Java source code to reading a output file from HDFS. The idea of this exercise is to get you started with Hadoop and the MapReduce concept. You may seen the WordCount from Hadoop official website or documentation and this example has a same functions (Map/Reduce) except that you will be computing the basic statistics such as min, max, average, and standard deviation of a given data set.

The input to the program will be a text file(s) carrying exactly one floating point number per line. The result file includes min, max, average, and standard deviation.

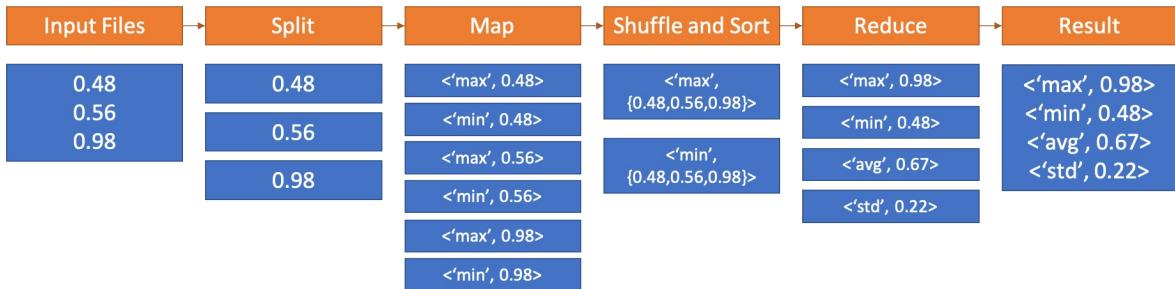


Figure 105: MapReduce example in Docker

#### 13.3.3.5.1.1 Base Location

The example is available within the container at:

```
container$ cd /cloudmesh/examples/statistics
```

#### 13.3.3.5.1.2 Input Files

A test input files are available under `/cloudmesh/examples/statistics/input_data` directory inside of the container. The statistics values for this input are Min: 0.20 Max: 19.99 Avg: 9.51 StdDev: 5.55 for all input files.

10 files contain 55000 lines to process and each line is a random float point value ranging from 0.2 to 20.0.

#### 13.3.3.5.1.3 Compilation

The source code file name is `MinMaxAvgStd.java` which is available at `/cloudmesh/examples/statistics/src`.

There are three functions in the code Map, Reduce and Main where Map reads each line of a file and updates values to calculate minimum, maximum values and Reduce collects mappers to produce average and standard deviation values at last.

```
$ export HADOOP_CLASSPATH=`$HADOOP_HOME/bin/hadoop classpath`
$ mkdir /cloudmesh/examples/statistics/dest
$ javac -classpath $HADOOP_CLASSPATH -d /cloudmesh/examples/statistics/dest /cloudmesh/exar
```

These commands simply prepare compiling the example code and the compiled class files are generated at the dest location.

#### 13.3.3.5.1.4 Archiving Class Files

Jar command tool helps archiving classes in a single file which will be used when Hadoop runs this example. This is useful because a jar file contains all necessary files to run a program.

```
$ cd /cloudmesh/examples/statistics
$ jar -cvf stats.jar -C ./dest/ .
```

#### 13.3.3.5.1.5 HDFS for Input/Output

The input files need to be uploaded to HDFS as Hadoop runs this example by reading input files from HDFS.

```
$ export PATH=$PATH:/HADOOP_HOME/bin
$ hadoop fs -mkdir stats_input
$ hadoop fs -put input_data/* stats_input
$ hadoop fs -ls stats_input/
```

If uploading is completed, you may see file listings like:

```
Found 10 items
-rw-r--r-- 1 root supergroup 13942 2018-02-28 23:16 stats_input/data_1000.txt
-rw-r--r-- 1 root supergroup 139225 2018-02-28 23:16 stats_input/data_10000.txt
-rw-r--r-- 1 root supergroup 27868 2018-02-28 23:16 stats_input/data_2000.txt
-rw-r--r-- 1 root supergroup 41793 2018-02-28 23:16 stats_input/data_3000.txt
-rw-r--r-- 1 root supergroup 55699 2018-02-28 23:16 stats_input/data_4000.txt
-rw-r--r-- 1 root supergroup 69663 2018-02-28 23:16 stats_input/data_5000.txt
-rw-r--r-- 1 root supergroup 83614 2018-02-28 23:16 stats_input/data_6000.txt
-rw-r--r-- 1 root supergroup 97490 2018-02-28 23:16 stats_input/data_7000.txt
-rw-r--r-- 1 root supergroup 111451 2018-02-28 23:16 stats_input/data_8000.txt
-rw-r--r-- 1 root supergroup 125337 2018-02-28 23:16 stats_input/data_9000.txt
```

#### 13.3.3.5.1.6 Run Program with a Single Input File

We are ready to run the program to calculate values from text files. First, we simply run the program with a single input file to see how it works. `data_1000.txt` contains 1000 lines of floats, we use this file here.

```
$ hadoop jar stats.jar exercise.MinMaxAvgStd stats_input/data_1000.txt stats_output_1000
```

The command runs with input parameters which indicate a jar file (the program, stats.jar), `exercise.MinMaxAvgStd` (package name.class name), input file path (`stats_input/data_1000.txt`) and output file path (`stats_output_1000`).

The sample results that the program produces look like this:

```
18/02/28 23:48:50 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/02/28 23:48:50 INFO input.FileInputFormat: Total input paths to process: 1
18/02/28 23:48:50 INFO mapreduce.JobSubmitter: number of splits:1
18/02/28 23:48:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1519877569596
18/02/28 23:48:51 INFO impl.YarnClientImpl: Submitted application application_1519877569596
18/02/28 23:48:51 INFO mapreduce.Job: The url to track the job: http://f5e82d68ba4a:8088/proxy/application_1519877569596_0002
18/02/28 23:48:51 INFO mapreduce.Job: Running job: job_1519877569596_0002
18/02/28 23:48:56 INFO mapreduce.Job: Job job_1519877569596_0002 running in uber mode: false
18/02/28 23:48:56 INFO mapreduce.Job: map 0% reduce 0%
18/02/28 23:49:00 INFO mapreduce.Job: map 100% reduce 0%
18/02/28 23:49:05 INFO mapreduce.Job: map 100% reduce 100%
18/02/28 23:49:05 INFO mapreduce.Job: Job job_1519877569596_0002 completed successfully
18/02/28 23:49:05 INFO mapreduce.Job: Counters: 49
 File System Counters
 FILE: Number of bytes read=81789
 FILE: Number of bytes written=394101
 FILE: Number of read operations=0
 FILE: Number of large read operations=0
 FILE: Number of write operations=0
 HDFS: Number of bytes read=14067
 HDFS: Number of bytes written=86
 HDFS: Number of read operations=6
 HDFS: Number of large read operations=0
 HDFS: Number of write operations=2
 Job Counters
 Launched map tasks=1
 Launched reduce tasks=1
 Data-local map tasks=1
 Total time spent by all maps in occupied slots (ms)=2107
 Total time spent by all reduces in occupied slots (ms)=2316
 Total time spent by all map tasks (ms)=2107
 Total time spent by all reduce tasks (ms)=2316
 Total vcore-seconds taken by all map tasks=2107
 Total vcore-seconds taken by all reduce tasks=2316
 Total megabyte-seconds taken by all map tasks=2157568
 Total megabyte-seconds taken by all reduce tasks=2371584
 Map-Reduce Framework
 Map input records=1000
 Map output records=3000
 Map output bytes=75783
 Map output materialized bytes=81789
 Input split bytes=125
 Combine input records=0
 Combine output records=0
 Reduce input groups=3
 Reduce shuffle bytes=81789
 Reduce input records=3000
 Reduce output records=4
 Spilled Records=6000
 Shuffled Maps =1
 Failed Shuffles=0
 Merged Map outputs=1
 GC time elapsed (ms)=31
 CPU time spent (ms)=1440
 Physical memory (bytes) snapshot=434913280
 Virtual memory (bytes) snapshot=1497260032
```

```
Total committed heap usage (bytes)=402653184
Shuffle Errors
 BAD_ID=0
 CONNECTION=0
 IO_ERROR=0
 WRONG_LENGTH=0
 WRONG_MAP=0
 WRONG_REDUCE=0
File Input Format Counters
 Bytes Read=13942
File Output Format Counters
 Bytes Written=86
```

The second line of the following logs indicates that the number of input files is 1.

#### 13.3.3.5.1.7 Result for Single Input File

We reads results from HDFS by:

```
$ hadoop fs -cat stats_output_1000/part-r-00000
```

The sample output looks like:

```
Max: 19.9678704297
Min: 0.218880718983
Avg: 10.225467263249385
Std: 5.679809322880863
```

#### 13.3.3.5.1.8 Run Program with Multiple Input Files

The first run was done pretty quickly (1440 milliseconds took according to the sample result above) because the input file size was small (1,000 lines) and it was a single file. We provides more input files with a larger size (2,000 to 10,000 lines). Input files are already uploaded to HDFS. We simply run the program again with a slight change in the parameters.

```
$ hadoop jar stats.jar exercise.MinMaxAvgStd stats_input/ stats_output_all
```

The command is almost same except that an input path is a directory and a new output directory. Note that every time that you run this program, the output directory will be created which means that you have to provide a new directory name unless you delete it.

The sample output messages look like the following which is almost identical compared to the previous run except that this time the number of input files to process is 10, see the line two below:

```
18/02/28 23:17:18 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
18/02/28 23:17:18 INFO input.FileInputFormat: Total input paths to process: 10
18/02/28 23:17:18 INFO mapreduce.JobSubmitter: number of splits:10
18/02/28 23:17:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1519877569596
18/02/28 23:17:19 INFO impl.YarnClientImpl: Submitted application application_1519877569596
18/02/28 23:17:19 INFO mapreduce.Job: The url to track the job: http://f5e82d68ba4a:8088/p...
18/02/28 23:17:19 INFO mapreduce.Job: Running job: job_1519877569596_0001
18/02/28 23:17:24 INFO mapreduce.Job: Job job_1519877569596_0001 running in uber mode: false
18/02/28 23:17:24 INFO mapreduce.Job: map 0% reduce 0%
18/02/28 23:17:32 INFO mapreduce.Job: map 40% reduce 0%
18/02/28 23:17:33 INFO mapreduce.Job: map 60% reduce 0%
18/02/28 23:17:36 INFO mapreduce.Job: map 70% reduce 0%
18/02/28 23:17:37 INFO mapreduce.Job: map 100% reduce 0%
18/02/28 23:17:39 INFO mapreduce.Job: map 100% reduce 100%
18/02/28 23:17:39 INFO mapreduce.Job: Job job_1519877569596_0001 completed successfully
18/02/28 23:17:39 INFO mapreduce.Job: Counters: 49
 File System Counters
 FILE: Number of bytes read=4496318
 FILE: Number of bytes written=10260627
 FILE: Number of read operations=0
 FILE: Number of large read operations=0
 FILE: Number of write operations=0
 HDFS: Number of bytes read=767333
 HDFS: Number of bytes written=84
 HDFS: Number of read operations=33
 HDFS: Number of large read operations=0
 HDFS: Number of write operations=2
 Job Counters
 Launched map tasks=10
 Launched reduce tasks=1
 Data-local map tasks=10
 Total time spent by all maps in occupied slots (ms)=50866
 Total time spent by all reduces in occupied slots (ms)=4490
 Total time spent by all map tasks (ms)=50866
 Total time spent by all reduce tasks (ms)=4490
 Total vcore-seconds taken by all map tasks=50866
 Total vcore-seconds taken by all reduce tasks=4490
 Total megabyte-seconds taken by all map tasks=52086784
 Total megabyte-seconds taken by all reduce tasks=4597760
 Map-Reduce Framework
 Map input records=55000
 Map output records=165000
 Map output bytes=4166312
 Map output materialized bytes=4496372
 Input split bytes=1251
 Combine input records=0
 Combine output records=0
 Reduce input groups=3
 Reduce shuffle bytes=4496372
 Reduce input records=165000
 Reduce output records=4
 Spilled Records=330000
 Shuffled Maps =10
```

```
Failed Shuffles=0
Merged Map outputs=10
GC time elapsed (ms)=555
CPU time spent (ms)=16040
Physical memory (bytes) snapshot=2837708800
Virtual memory (bytes) snapshot=8200089600
Total committed heap usage (bytes)=2213019648
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=766082
File Output Format Counters
Bytes Written=84
```

#### 13.3.3.5.1.9 Result for Multiple Files

```
$ hadoop fs -cat stats_output_all/part-r-00000
```

The expected result looks like:

```
Max: 19.999191254
Min: 0.200268613863
Avg: 9.514884854468903
Std: 5.553921579413547
```

#### 13.3.3.5.2 Conclusion

The example program of calculating some values by reading multiple files shows how Map/Reduce is written by a Java programming language and how Hadoop runs its program using HDFS. We also observed the one of benefits using Docker container which is that the hassle of configuration and installation of Hadoop is not necessary anymore.

#### 13.3.3.6 References

- The details of the new version is available from the official site at <http://hadoop.apache.org/docs/r3.0.3/index.html>

#### 13.3.4 DOCKER PAGERANK



PageRank is a popular example algorithm used to display the ability of big data applications to run parallel tasks. This example will show how the docker hadoop image can be used to execute the Pagerank example which is available in `/cloudmesh/examples/pagerank`

### 13.3.4.1 Use the automated script

We make the steps of compiling java source, archiving class files, load input files and run the program into one single script. To execute it with the input file: `PageRankDataGenerator/pagerank5000g50.input.0`, using 5000 urls and 1 iteration:

```
$ cd /cloudmesh/examples/pagerank
$./compileAndExecHadoopPageRank.sh PageRankDataGenerator/pagerank5000g50.input.0 5000 1
```

Result will look like

```
output.pagerank/part-r-00000
```

The head of the result will look like

```
head output.pagerank/part-r-00000
```

```
0 2.9999999999999997E-5
1 2.9999999999999997E-5
2 2.9999999999999997E-5
3 2.9999999999999997E-5
4 2.9999999999999997E-5
5 2.9999999999999997E-5
6 2.9999999999999997E-5
7 2.9999999999999997E-5
8 2.9999999999999997E-5
9 2.9999999999999997E-5
```

### 13.3.4.2 Compile and run by hand

If one wants to generate the java class files and archive them as the previous exercise, one could use the following code (which is actually inside `compileAndExecHadoopPageRank.sh`)

```
export HADOOP_CLASSPATH=`$HADOOP_PREFIX/bin/hadoop classpath`
mkdir /cloudmesh/examples/pagerank/dist
$ find /cloudmesh/examples/pagerank/src/indiana/cgl/hadoop/pagerank/ \
```

```
-name "*.java" | xargs javac -classpath $HADOOP_CLASSPATH \
-d /cloudmesh/examples/pagerank/dist
$ cd /cloudmesh/examples/pagerank/dist
$ jar -cvf HadoopPageRankMooc.jar -C . .
```

## Load input files to HDFS

```
$ export PATH=$PATH:$HADOOP_PREFIX/bin
$ cd /cloudmesh/examples/pagerank/
$ hadoop fs -mkdir input.pagerank
$ hadoop fs -put PageRankDataGenerator/pagerank5000g50.input.0 input.pagerank
```

- Run program with the [PageRank Inputs File Directory] [PageRank Output Directory][Number of URLs][Number Of Iterations]

```
$ hadoop jar dist/HadoopPageRankMooc.jar indiana.cg1.hadoop.pagerank.HadoopPageRank input.pagerank
```

## Result

```
$ hadoop fs -cat output.pagerank/part-r-00000
```

## 13.3.5 APACHE SPARK WITH DOCKER



### 13.3.5.1 Pull Image from Docker Repository

We use a Docker image from Docker Hub: (<https://hub.docker.com/r/sequenceiq/spark/>) This repository contains a Docker file to build a Docker image with Apache Spark and Hadoop Yarn.

```
$ docker pull sequenceiq/spark:1.6.0
```

### 13.3.5.2 Running the Image

In this step, we will launch a Spark container.

#### 13.3.5.2.1 Running interactively

```
$ docker run -it -p 8088:8088 -p 8042:8042 -h sandbox sequenceiq/spark:1.6.0 bash
```

#### 13.3.5.2.2 Running in the background

```
$ docker run -d -h sandbox sequenceiq/spark:1.6.0 -d
```

### 13.3.5.3 Run Spark

After a container is launched, we can run Spark in the following two modes: (1) yarn-client and (2) yarn-cluster. The differences between the two modes can be found here: <https://spark.apache.org/docs/latest/running-on-yarn.html>

#### 13.3.5.3.1 Run Spark in Yarn-Client Mode

```
$ spark-shell --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1
```

#### 13.3.5.3.2 Run Spark in Yarn-Cluster Mode

```
$ spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1 $SPARK_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar 10
```

### 13.3.5.4 Observe Task Execution from Running Logs of SparkPi

Let us observe Spark task execution by adjusting the parameter of SparkPi and the Pi result from the following two commands.

```
$ spark-submit --class org.apache.spark.examples.SparkPi \
 --master yarn-client --driver-memory 1g \
 --executor-memory 1g \
 --executor-cores 1 $SPARK_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar 10
$ spark-submit --class org.apache.spark.examples.SparkPi \
 --master yarn-client --driver-memory 1g \
 --executor-memory 1g \
 --executor-cores 1 $SPARK_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar 10000
```

### 13.3.5.5 Write a Word-Count Application with Spark RDD

Let us write our own word-count with Spark RDD. After the shell has been started, copy and paste the following code in console line by line.

#### 13.3.5.5.1 Launch Spark Interactive Shell

```
$ spark-shell --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1
```

### 13.3.5.5.2 Program in Scala

```
val textFile = sc.textFile("file:///etc/hosts")
val words = textFile.flatMap(line => line.split("\\s+"))
val counts = words.map(word => (word, 1)).reduceByKey(_ + _)
counts.values.sum()
```

### 13.3.5.5.3 Launch PySpark Interactive Shell

```
$ pyspark --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1
```

### 13.3.5.5.4 Program in Python

```
textFile = sc.textFile("file:///etc/hosts")
words = textFile.flatMap(lambda line:line.split())
counts = words.map(lambda word:(word, 1)).reduceByKey(lambda x,y: x+y)
counts.map(lambda x:x[1]).sum()
```

## 13.3.5.6 Docker Spark Examples

### 13.3.5.6.1 K-Means Example

First we need to pull the image from the Docker Hub :

```
$ docker pull sequenceiq/spark-native-yarn
```

It will take sometime to download the image. Now we have to run docker spark image interactively.

```
$ docker run -i -t -h sandbox sequenceiq/spark-native-yarn /etc/bootstrap.sh -bash
```

This will take you to the interactive mode.

Let's run a sample KMeans example. This is already built with Spark.

Here we specify the data data set from a local folder inside the image and we run the sample class KMeans in the sample package. The sample data set used is inside the sample-data folder. Spark has it's own format for machine learning datasets. Here the kmeans\_data.txt file contains the KMeans dataset.

```
$./bin/spark-submit --class sample.KMeans \
--master execution-context:org.apache.spark.tez.TezJobExecutionContext
\
```

```
--conf update-classpath=true \
./lib/spark-native-yarn-samples-1.0.jar /sample-data/kmeans_data.txt
```

If you run this successfully, you can get an output as shown here.

```
Finished iteration (delta = 0.0)
Final centers:
DenseVector(0.1500000000000002, 0.1500000000000002, 0.1500000000000002)
DenseVector(9.2, 9.2, 9.2)
DenseVector(0.0, 0.0, 0.0)
DenseVector(9.05, 9.05, 9.05)
```

### 13.3.5.6.2 Join Example

Run the following command to do a sample join operation on a given dataset. Here we use two datasets, namely join1.txt and join2.txt. Then we perform the join operation that we discussed in the theory section.

```
$./bin/spark-submit --class sample.Join --master execution-context:org.apache.spark.tez.TezMaster
```

### 13.3.5.6.3 Word Count

In this example the wordcount.txt will used to do the word count using multiple reducers. Number 1 at the end of the command determines the number of reducers. As spark can run multiple reducers, we can specify the number as a parameter to the programme.

```
$./bin/spark-submit --class sample.WordCount --master execution-context:org.apache.spark.local
```

### 13.3.5.7 Interactive Examples

Here we need a new image to work on. Let's run the following command. This will pull the necessary repositories from docker hub, as we do not have most of the dependencies related to it. This can take a few minutes to download everything.

```
$ docker run -it-p 8888:8888 -v $PWD:/cloudmesh/spark --name spark jupyter/pyspark-notebook
```

Here you will get the following output in the terminal.

```
docker run -it -p 8888:8888 -v $PWD:/cloudmesh/spark --name spark jupyter/pyspark-notebook
Unable to find image 'jupyter/pyspark-notebook:latest' locally
latest: Pulling from jupyter/pyspark-notebook
a48c500ed24e: Pull complete
1e1de00ff7e1: Pull complete
0330ca45a200: Pull complete
471db38bcfbf: Pull complete
0b4aba487617: Pull complete
d44ea0cd796c: Pull complete
5ac827d588be: Pull complete
d8d7747a335e: Pull complete
08790511e3e9: Pull complete
e3c68aea9a5f: Pull complete
484c6d5fc38a: Pull complete
0448c1360cb9: Pull complete
61d7e6dc705d: Pull complete
92f1091ed72b: Pull complete
8045d3663a7e: Pull complete
1bde7ba25439: Pull complete
5618f8ed38b4: Pull complete
f08523cb6144: Pull complete
99eee56fd2f: Pull complete
b37b1ce39785: Pull complete
aee4b9eac4ea: Pull complete
f810ef87439d: Pull complete
038786dce388: Pull complete
ded31312ea33: Pull complete
30221ffdd1a6: Pull complete
da1d368f8592: Pull complete
523809a30a21: Pull complete
47ab1b230dd2: Pull complete
442f9435e1a9: Pull complete
Digest: sha256:f8b6309cd39481de1a169143189ed0879b12b56fe286d254d03fa34ccad90734
Status: Downloaded newer image for jupyter/pyspark-notebook:latest
Container must be run with group "root" to update passwd file
Executing the command: jupyter notebook
[I 15:47:52.900 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[I 15:47:53.167 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.6/site-packages/jupyterlab
[I 15:47:53.167 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 15:47:53.176 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 15:47:53.177 NotebookApp] The Jupyter Notebook is running at:
[I 15:47:53.177 NotebookApp] http://(3a3d9f7e2565 or 127.0.0.1):8888/?token=f22492fe7ab8206ac2223359e0603a0dff5
[I 15:47:53.177 NotebookApp] Use Control-C to stop this server and shut down all kernels (1 active)
[C 15:47:53.177 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://(3a3d9f7e2565 or 127.0.0.1):8888/?token=f22492fe7ab8206ac2223359e0603a0dff5
```

Please copy the url shown at the end of the terminal output and go to that url in the browser.

You will see the following output in the browser, (Use Google

Chrome)



## Jupyter Notebook in Browser

First navigate to the work folder. Let us create a new python file here. Click python3 in the new menu.



## Create a new python file

Now add the following content in the new file. In Jupyter notebook, you can enter a python command or python code and press

SHIFT + ENTER

This will run the code interactively.

Now let's create the following content.

```
import os
os.getcwd()

import pyspark
sc = pyspark.SparkContext('local[*]')
rdd = sc.parallelize(range(1000))
rdd.takeSample(False, 5)
```

Now let us do the following.

In the following stage we configure spark context and import the necessary files.

```

os.makedirs("data")

from pyspark.mllib.clustering import KMeans, KMeansModel
from numpy import array
from math import sqrt
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.linalg import SparseVector
sc.version

```

Next stage we use sample data set by creating them in form of an array and we train the kmeans algorithm.

```

sparse_data = [
 SparseVector(3, {1:1.0}),
 SparseVector(3, {1:1.1}),
 SparseVector(3, {2:1.0}),
 SparseVector(3, {2:1.1})
]

model = KMeans.train(sc.parallelize(sparse_data), 2, initializationMode='k-means||',
 seed=50, initializationSteps=5, epsilon=1e-4)

model.predict(array([0.,1.,0.]))
model.predict(array([0.,0.,1.]))
model.predict(sparse_data[0])
model.predict(sparse_data[2])

```

In the final stage we put sample values and check the predictions on the cluster. In addition to that feed the data using SparseVector format and we add the kmeans initialization mode, the error margin and the palatalization. We put the step size as 5 for this example. In the previous one we did not specify any parameters.

The predict term predicts the cluster id which it belongs to.

```

data = array([0.0, 0.0, 1.0, 1.0, 9.0, 8.0, 8.0, 9.0]).reshape(4, 2)
model = KMeans.train(sc.parallelize(data), 2, initializationMode='random',
 seed=50, initializationSteps=5, epsilon=1e-4)
model.predict(array([0.0, 0.0])) == model.predict(array([1.0, 1.0]))
model.predict(array([8.0, 9.0]))
model.predict(array([8.0, 9.0])) == model.predict(array([9.0, 8.0]))
model.k
model.computeCost(sc.parallelize(data))

```

Then in the following way you can check whether two data points belong to one cluster or not.

```
isinstance(model.clusterCenters, list)
```

### 13.3.5.7.1 Stop Docker Container

```
$ docker stop spark
```

### 13.3.5.7.2 Start Docker Container Again

```
$ docker start spark
```

### 13.3.5.7.3 Remove Docker Container

```
$ docker rm spark
```

## 13.4 KUBERNETES



### 13.4.1 INTRODUCTION TO KUBERNETES



#### ◆ Learning Objectives

- What is Kubernetes?
- What are containers?
- Cluster components in Kubernetes
- Basic Units in Kubernetes
- Run an example with Minikube
- Interactive online tutorial
- Have a solid understanding of Containers and Kubernetes
- Understand the Cluster components of Kubernetes
- Understand the terminology of Kubernetes
- Gain practical experience with kubernetes
- With minikube
- With an interactive online tutorial

---

Kubernetes is an open-source platform designed to automate deploying, scaling, and operating application containers.

- <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/>

## [kubernetes/](#)

With Kubernetes, you can:

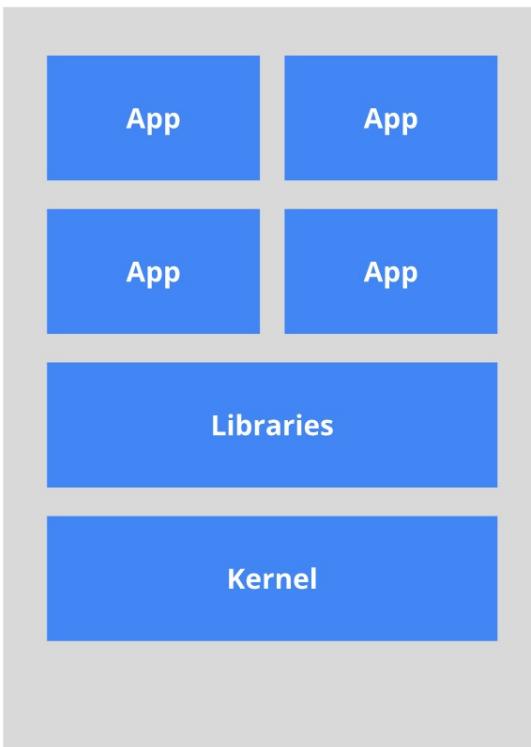
- Deploy your applications quickly and predictably.
- Scale your applications on the fly.
- Roll out new features seamlessly.
- Limit hardware usage to required resources only.
- Run applications in public and private clouds.

Kubernetes is

- Portable: public, private, hybrid, multi-cloud
- Extensible: modular, pluggable, hookable, composable
- Self-healing: auto-placement, auto-restart, auto-replication, auto-scaling

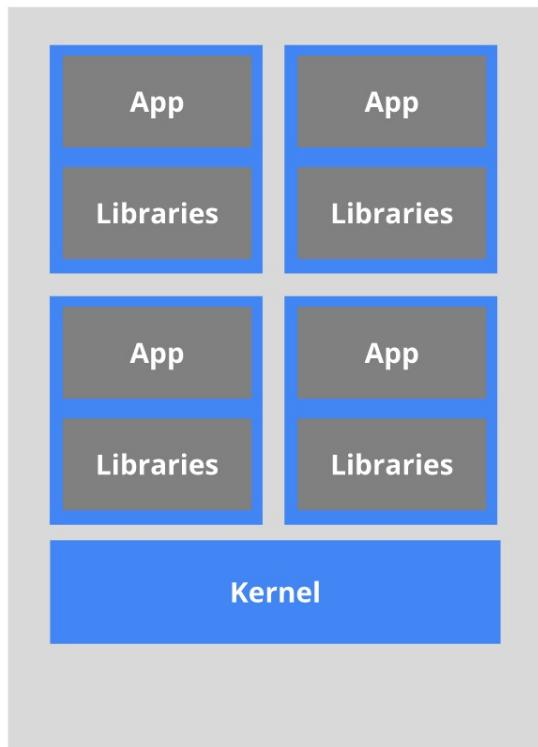
### **13.4.1.1 What are containers?**

**The old way:** Applications on host



*Heavyweight, non-portable  
Relies on OS package manager*

**The new way:** Deploy containers



*Small and fast, portable  
Uses OS-level virtualization*

Figure 106: Kubernetes Containers [\[Image Source\]](#)

[Figure 106](#) shows a depiction of the container architecture.

### 13.4.1.2 Terminology

In kubernetes we are using the following terminology

Pods:

A pod (as in a pod of whales or pea pod) is a group of one or more containers (such as Docker containers), with shared storage/network, and a specification for how to run the containers. A pod's contents are always co-located and co-scheduled, and run in a shared context. A pod models an application-specific logical host. It contains one or more application containers which are relatively tightly coupled. In a pre-container world, they would have executed on the same

physical or virtual machine.

Services:

Service is an abstraction which defines a logical set of Pods and a policy by which to access them. Sometimes they are called a micro-service. The set of Pods targeted by a Service is (usually) determined by a Label Selector.

Deployments:

A Deployment controller provides declarative updates for Pods and ReplicaSets. You describe a desired state in a Deployment object, and the Deployment controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

### **13.4.1.3 Kubernetes Architecture**

The architecture of kubernets is shown in [Figure 107](#).

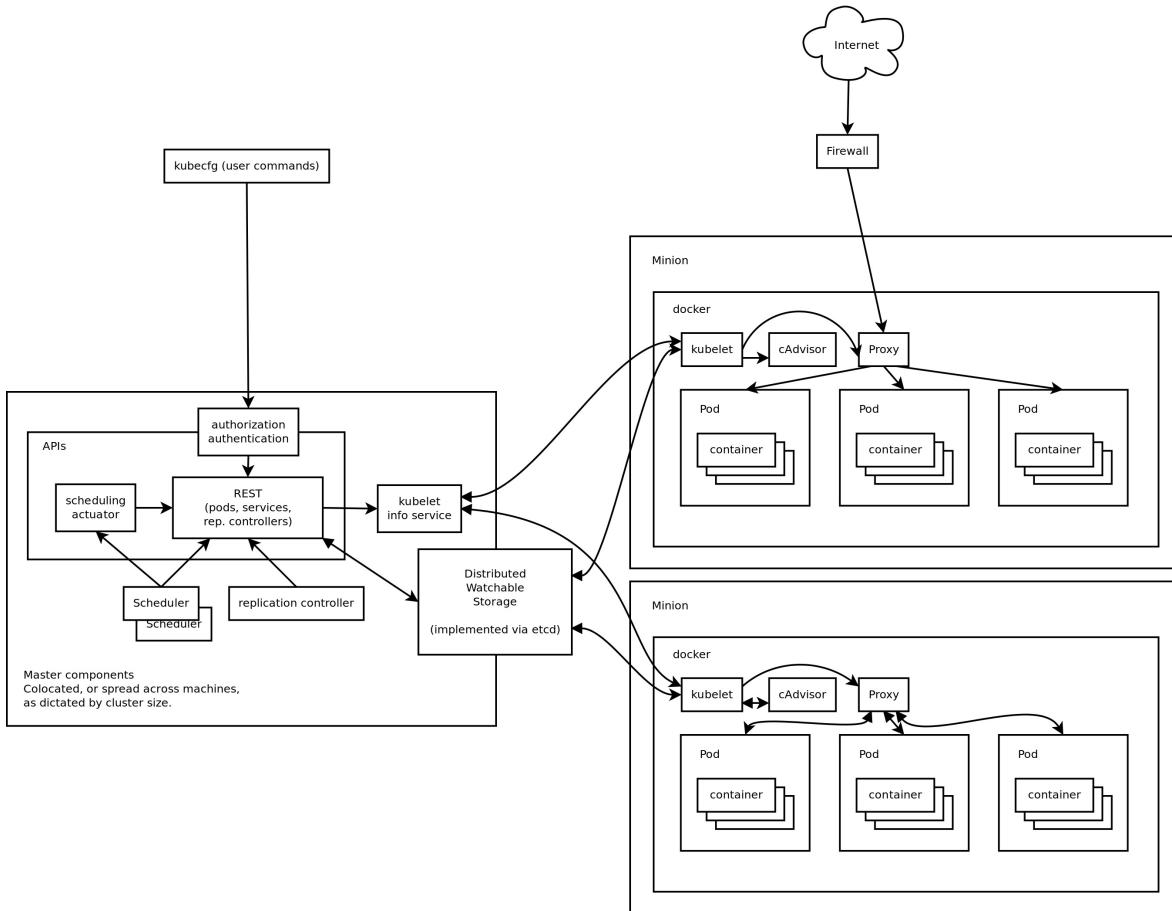


Figure 107: Kubernetes (Source: Google)

### 13.4.1.4 Minikube

To try out kubernetes on your own computer you can download and install minikube. It deploys and runs a single-node Kubernetes cluster inside a VM. Hence it provide a reasonable environment not only to try it out, but also for development [\[cite\]](#).

In this section we will first discuss how to install minikube and then showcase an example.

#### 13.4.1.4.1 Install minikube

##### 13.4.1.4.1.0.1 OSX

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/minikube-darw:
```

### 13.4.1.4.1.0.2 Windows 10

We assume that you have installed Oracle VirtualBox in your machine which must be a version 5.x.x.

Initially, we need to download two executables.

[Download Kubectl](#)

[Download Minikube](#)

After downloading these two executables place them in the cloudmesh directory we earlier created. Rename the `minikube-windows-amd64.exe` to `minikube.exe`. Make sure minikube.exe and kubectl.exe lie in the same directory.

### 13.4.1.4.1.0.3 Linux

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/minikube-linux-amd64
```

Installing KVM2 is important for Ubuntu distributions

```
$ sudo apt install libvirt-bin qemu-kvm
$ sudo usermod -a -G libvирtd $(whoami)
$ newgrp libvирtd
```

We are going to run minikube using KVM2 libraries instead of virtualbox libraries for windows installation.

Then install the drivers for KVM2,

```
$ curl -LO https://storage.googleapis.com/minikube/releases/latest/docker-machine-driver-kvm
```

#### 13.4.1.4.2 Start a cluster using Minikube

##### 13.4.1.4.2.0.1 OSX Minikube Start

```
$ minikube start
```

##### 13.4.1.4.2.0.2 Ubuntu Minikube Start

```
$ minikube start --vm-driver=kvm2
```

### 13.4.1.4.2.0.3 Windows 10 Minikube Start

In this case you must run Windows PowerShell as administrator. For this search for the application in search and right click and click Run as administrator. If you are an administrator it will run automatically but if you are not please make sure you provide the admin login information in the pop up.

```
$ cd C:\Users\<username>\Documents\cloudmesh
$.\minikube.exe start --vm-driver="virtualbox"
```

#### 13.4.1.4.3 Create a deployment

```
$ kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --port=8080
```

#### 13.4.1.4.4 Expose the service

```
$ kubectl expose deployment hello-minikube --type=NodePort
```

#### 13.4.1.4.5 Check running status

This step is to make sure you have a pod up and running.

```
$ kubectl get pod
```

#### 13.4.1.4.6 Call service api

```
$ curl $(minikube service hello-minikube --url)
```

#### 13.4.1.4.7 Take a look from Dashboard

```
$ minikube dashboard
```

If you want to get an interactive dashboard,

```
$ minikube dashboard --url=true
http://192.168.99.101:30000
```

Browse to <http://192.168.99.101:30000> in your web browser and it will provide a GUI dashboard regarding minikube.

#### **13.4.1.4.8 Delete the service and deployment**

```
$ kubectl delete service hello-minikube
$ kubectl delete deployment hello-minikube
```

#### **13.4.1.4.9 Stop the cluster**

For all platforms we can use the following command.

```
$ minikube stop
```

#### **13.4.1.5 Interactive Tutorial Online**

- Start cluster <https://kubernetes.io/docs/tutorials/kubernetes-basics/cluster-interactive/>
- Deploy app <https://kubernetes.io/docs/tutorials/kubernetes-basics/cluster-interactive>
- Explore <https://kubernetes.io/docs/tutorials/kubernetes-basics/explore-intro/>
- Expose <https://kubernetes.io/docs/tutorials/kubernetes-basics/expose-intro/>
- Scale <https://kubernetes.io/docs/tutorials/kubernetes-basics/scale-intro/>
- Update <https://kubernetes.io/docs/tutorials/kubernetes-basics/update-interactive/>
- MiniKube <https://kubernetes.io/docs/tutorials/stateless-application/hello-minikube/>

### **13.4.2 USING KUBERNETES ON FUTURESYSTEMS**



This section introduces you on how to use the Kubernetes cluster on FutureSystems. Currently we have deployed kubernetes on our cluster called echo.

#### **13.4.2.1 Getting Access**

You will need an account on FutureSystems and upload the ssh key to the FutureSystems portal from the computer from which you want to

login to echo. To verify, if you have access try to see if you can log into victor.futuresystems.org. You need to be a member of a valid FutureSystems project.

For Fall 2018 classes at IU you need to be in the following project:

<https://portal.futuresystems.org/project/553>

If you have verified that you have access to the victor, you can now try to login to the kubernetes cluster head node with the same username and key. Run these first on your **local machine** to set the username and login host:

```
$ export ECHOK8S=149.165.150.85
$ export FS_USER=<put your futersystem account name here>
```

Then you can login to the kubernetes head node by running:

```
$ ssh $FS_USER@$ECHOK8S
```

**NOTE: If you have access to victor but not the kubernetes system, your project may not have been authorized to access the kubernetes cluster. Send a ticket to FutureSystems ticket system to request this.**

Once you are logged in to the kubernetes cluster head node you can run commands on the **remote echo kubernetes machine** (all commands below except stated otherwise) to use the kubernetes installation there. First try to run:

```
$ kubectl get pods
```

This will let you know if you have access to kubernetes and verifies if the kubectl command works for you. Naturally it will also list the pods.

### 13.4.2.2 Example Use

The following command runs an image called Nginx with two replicas, Nginx is a popular web sever which is well known as a high

performance load balancer.

```
$ kubectl run nginx --replicas=2 --image=nginx --port=80
```

As a result of this one deployment was created, and two PODs are created and started. If you encounter an error stating that the deployment already exists when executing the previous command that is because the command has already been executed. To see the deployment, please use the command, this command should work even if you noticed the error mentioned.

```
$ kubectl get deployment
```

This will result in the following output

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
nginx	2	2	2	2	7m

To see the pods please use the command

```
$ kubectl get pods
```

This will result in the following output

NAME	READY	STATUS	RESTARTS	AGE
nginx-7587c6fdb6-4jnh6	1/1	Running	0	7m
nginx-7587c6fdb6-pxpsz	1/1	Running	0	7m

If we want to see more detailed information we can use the command

```
$ kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
nginx-75...-4jnh6	1/1	Running	0	8m	192.168.56.2	e003
nginx-75...-pxpsz	1/1	Running	0	8m	192.168.255.66	e005

Please note the IP address field. Make sure you are using the IP address that is listed when you execute the command since the IP address may have changed. Now if we try to access the nginx homepage with wget (or curl)

```
$ wget 192.168.56.2
```

we see the following output:

```
--2018-02-20 14:05:59-- http://192.168.56.2/
Connecting to 192.168.56.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 612 [text/html]
Saving to: 'index.html'

index.html 100%[=====] 612 ---KB/s in 0s

2018-02-20 14:05:59 (38.9 MB/s) - 'index.html' saved [612/612]
```

It verifies that the specified image was running, and it is accessible from within the cluster.

Next we need to start thinking about how we access this web server from outside the cluster. We can explicitly exposing the service with the following command. You can change the name that is set using `--name` to what you want. Given that is adheres to the naming standards. If the name you enter is already in the system your command will return an error saying the service already exists.

```
$ kubectl expose deployment nginx --type=NodePort --name=abc-nginx-ext
```

We will see the response

```
$ service "nginx-external" exposed
```

To find the exposed ip addresses, we simply issue the command

```
$ kubectl get svc
```

We see something like this

NAME	TYPE	CLUSTER-IP	EXTERN PORT(S)	AGE
			AL-IP	
kubernetes	ClusterIP	10.96.0.1	<none> 443/TCP	8h
abc-nginx-ext	NodePort	10.110.177.35	<none> 80:31386/TCP	3s

please note that we have given a unique name.

---

For IU students:

You could use your username or if you use one of our classes your hid. The number part will typically be sufficient. For class users that do not use the hid in the name we will terminate all instances without notification. In addition, we like you explicitly to add “-ext” to every

container that is exposed to the internet. Naturally we want you to shut down such services if they are not in use. Failure to do so may result in termination of the service without notice, and in the worst case revocation of your privileges to use echo.

---

In our example you will find the port on which our service is exposed and remapped to. We find the port **31386** in the value **80:31386/TCP** in the ports column for the running container.

Now if we visit this URL, which is the public IP of the head node followed by the exposed port number, from a browser on **your local machine**

```
http://149.165.150.85:31386
```

you should see the 'Welcome to nginx' page.

Once you have done all the work needed using the service you can delete it using the following command.

```
$ kubectl delete service <service-name>
```

### 13.4.2.3 Exercises

E.Kubernetes.fs.1:

Explore more complex service examples.

E.Kubernetes.fs.2:

Explore constructing a complex web app with multiple services.

E.Kubernetes.fs.3:

Define a deployment with a yaml file declaratively.

## 13.5 EXERCISES



## E.Docker.Swarm.1: Documentation

Develop a section in the handbook that deploys a Docker Swarm cluster on a number of ubuntu machines. Note that this may actually be easier as docker and docker swarm are distributed with recent versions of ubuntu. Just in case we are providing a link to an effort we found to install docker swarm. However we have not checked it or identified if it is useful.

- <https://rominirani.com/docker-swarm-tutorial-b67470cf8872>

## E.Docker.Swarm.2: Google Compute Engine

Develop a section that deploys a Docker Swarm cluster on Google Compute Engine. Note that this may actually be easier as docker and docker swarm are distributed with recent versions of ubuntu. Just in case we are providing a link to an effort we found to install docker swarm. However we have not checked it or identified if it is useful.

- <https://rominirani.com/docker-swarm-on-google-compute-engine-364765b400ed>

## E.SingleNodeHadoop:

Setup a single node hadoop environment.

This includes:

- (a).Create a Dockerfile that deploys hadoop in a container
- (b).Develop sample applications and tests to test your cluster.  
You can use wordcount or similar.

you will find a comprehensive installation instruction that sets up a hadoop cluster on a single node at

- <https://hadoop.apache.org/docs/current/hadoop-project->

[dist/hadoop-common/SingleCluster.html](https://hadoop.apache.org/docs/r3.0.0/dist/hadoop-common/SingleCluster.html)

#### E.MultiNodeHadoop:

Setup a hadoop cluster in a distributed environment.

This includes:

- (a).Create docker compose and Dockerfiles that deploys hadoop in kubernetes
- (b).Develop sample applications and tests to test your cluster. You can use wordcount or similar.

you will find a comprehensive installation instruction that sets up a hadoop cluster in a distributed environment at

- <https://hadoop.apache.org/docs/r3.0.0/dist/hadoop-common/ClusterSetup.html>

You can use this set of instructions or identify other resources on the internet that allow the creation of a hadoop cluster on kubernetes. Alternatively you can use docker compose for this exercise.

#### E.SparkCluster: Documentation

Develop a high quality section that installs a spark cluster in kubernetes. Test your deployment on minikube and also on Futuresystems echo.

You may want to get inspired from the talk Scalable Spark Deployment using Kubernetes:

- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-1/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-2/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-3/>
- <http://blog.madhukaraphatak.com/scaling-spark-with->

[kubernetes-part-4/](#)

- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-5/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-6/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-7/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-8/>
- <http://blog.madhukaraphatak.com/scaling-spark-with-kubernetes-part-9/>

Make sure you do not plagiarize.



### 14.1 FaaS



#### 14.1.1 Introduction

FaaS or Function as a service is a new paradigm in cloud computing that is gaining popularity recently. FaaS is also known as Serverless Computing to some. While the name Serverless implies that no servers are involved this is not true. So FaaS would be a better term to describe this technology. FaaS is built around functions and events. Functions are generally stateless and are executed within isolated containers. The execution of the functions can be thought of as an event driven model. A program or application in FaaS consist of a set of functions and a set of events or triggers that invoke or activate those functions. A function activation can result in another function activation as a result.

Generally FaaS specifies a set of constraints on what a function can be. The constraints include storage constraints such as a maximum size for the function, maximum memory allowed, execution time, etc. The exact constraints differ from provider to provider. AWS Lambda is considered as one of first FaaS offerings. Now most cloud providers offer their own version of FaaS. Several popular FaaS providers are listed below.

#### 14.1.2 Serverless Computing

In Serverless Computing, servers are still there, its just that we dont need to manage them.

Another advantage of going serverless is that you no longer need to keep a server running all the time. The server suddenly appears when you need it, then disappears when you're done with it. Now you can think in terms of functions instead of servers, and all your business

logic can now live within these functions.

### 14.1.3 Faas provider

- AWS Lambda <https://aws.amazon.com/lambda/>
- Azure Functions <https://azure.microsoft.com/en-us/services/functions/>
- IBM Cloud Functions <https://console.bluemix.net/openwhisk/>
- Google Cloud Functions <https://cloud.google.com/functions/>
- Iron.io <https://www.iron.io/>
- Webtask.io <https://webtask.io/>

Other than the providers there are also several open source FaaS offerings that are available to be used. One of the most complete and popular open source option would be Apache OpenWhisk, which was developed by IBM and later open sourced. IBM currently deploys OpenWhisk in IBM cloud and offers it as a IBM Cloud functions.

- OpenWhisk <https://github.com/apache/incubator-openwhisk>
- Funktion <https://github.com/funktionio/funktion>
- Iron Functions <https://github.com/iron-io/functions>
- Kubeless <https://github.com/kubeless/kubeless>
- Fission <https://github.com/fission/fission>
- FaaS-netes <https://github.com/alexellis/faas-netes>

There are many articles and tutorials online that provide very good information regarding FaaS. Below are some such resources that provide introductions and some example usecase's of FaaS

### 14.1.4 Resources

- <https://stackify.com/function-as-a-service-serverless-architecture/>
- [https://en.wikipedia.org/wiki/Serverless\\_computing](https://en.wikipedia.org/wiki/Serverless_computing)
- <https://azure.microsoft.com/en-us/overview/serverless-computing/>
- <https://aws.amazon.com/serverless/>

- <https://aws.amazon.com/lambda/>
- <https://www.infoworld.com/article/3093508/cloud-computing/what-serverless-computing-really-means.html>
- <https://techbeacon.com/aws-lambda-serverless-apps-5-things-you-need-know-about-serverless-computing>
- <https://blog.alexellis.io/introducing-functions-as-a-service/>

## 14.1.5 Usage Examples

- <https://aws.amazon.com/solutions/case-studies/netflix-and-aws-lambda/>
- <https://blog.alexellis.io/first-faas-python-function/>

## 14.2 AWS LAMBDA



### ❖ Learning Objectives

- Learn about AWS lambda
- Try out AWS Lambda practically

AWS Lambda is considered as one of the earliest FaaS implementations made available to end users. AWS Lambda provides a rich set of features that can be leveraged by users to build programs and applications on top of the FaaS framework. AWS Lambda supports many event types that developers can use to orchestrate their FaaS applications. And in line with the FaaS model AWS Lambda only charges users for actual execution time of the functions. For example if you host and deploy a function on AWS Lambda and only execute it for 1 minute every day you will only be charged for the 1 minute execution time.

AWS does not share how the internals of AWS Lambda work in detail but as with any general FaaS framework it should be leveraging various container technologies underneath. You can get a better understanding on how the internals of a FaaS framework is organized

by looking at the [OpenWhisk Section](#)

### 14.2.1 AWS Lambda Features

AWS Lambda provides many features that allows the creation of an FaaS application to be straight forward. An FaaS application normally consist of a set of functions and a set of events that activate or invoke those functions. AWS Lambda supports several programing languages that allow developers to develop the function they require in any of the programming languages that are supported. The following list show the programming languages that are currently supported by AWS Lambda for function creation.

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Go

Other than the functions the most important requirement for a good FaaS framework is a rich set of function invocation methods which allow users to tie together different events that happen in the echo system with the FaaS application. In this regard AWS Lambda supports many event sources, mainly from the AWS echo system. AWS documentation provides a complete list of supported event sources at [AWS Lambda event sources](#). For example a developer can configure an function to be invoked when a S3 bucket is updated, or configure an function to be invoked based on inputs received by Amazon Alexa, etc.

○ use also bibtex

### 14.2.2 Understanding Function limitations

Before you start working on FaaS frameworks it is important to understand the limitations and restrictions that are applied to functions. The limits and restrictions discussed in the section are

applicable to most FaaS frameworks but the exact values may differ based on the FaaS vendor. However the reason for most of the limitations are to maintain performance requirements. We will discuss several major limits below. For a complete list of limits in AWS Lambda please refer to the limits documentation [AWS Lambda Function Limits](#)

- use also bibtex

#### **14.2.2.1 Execution Time**

AWS Lambda limits the execution length of a function. Currently it is set to 15 minutes but was set at 300s previously, so the function limits have and may change with time. Other FaaS providers have different values for execution time limits, but in general each FaaS provider does define a execution time limit

#### **14.2.2.2 Function size**

AWS Lambda also sets several memory and storage limits for functions. Currently the maximum memory allocated to a function is 3008MB and the maximum allocated storage space is 512MB. However it is good to keep in mind that monetary charge for function execution increases with the amount of memory that is specified for the function.

#### **14.2.3 Understanding the free Tier**

If new users want to experiment with AWS lambda, AWS does provide a free tier for AWS Lambda, which include 1 million function invocations per month. You can find a more detailed description of the free tier in the AWS docs [AWS Lambda Pricing](#).

- use also bibtex

#### **14.2.4 Writing your fist Lambda function**

With the GUI interface it is relatively easy to try out your first Serverless function with AWS Lambda. Please follow the steps defined at [How to run your first AWS Lambda function in the cloud](#) (this link does not exist any longer)

○ use also bibtex

### 14.2.5 AWS Lambda UseCases

AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of the Amazon Web Services. It is a computing service that runs code in response to events, runs the code that has been loaded into the system and automatically manages the computing resources required by that code. According to the Lambda product page

“AWS Lambda lets you run code without provisioning or managing servers.” [Aws](#) (○ reference to bibtex)

For example, one of the use-cases would be that everytime AWS Lambda could resize the picture, after it is uploaded onto AWS S3 system and rendered on different devices like phone, ipad or desktop. The event that triggers the Lambda function is the file being uploaded to S3. Lambda then executes the function of resizing the image. The Seattle Times uses the AWS Lambda to automatically resize the images.

One key point to note here is that Amazon charges only when the functions are executed. So, The Seattle Times is charged for this service only when the images are been resized. Lambda can be used for Analytics. So lets say, there has been a purchase of a house on zillow, this data can be saved into a NoSQL database and this entry into the database is an event which can trigger Lambda function to load the order information into Amazon Redshift. Then we can run Analytics on top of this data. We can also build serverless applications composed of functions that are triggered by events and automatically deploy them using AWS CodePipeline and AWS CodeBuild. For more information, see Deploying Lambda-based Applications.

There are development groups or companies mainly startups, where they want to just focus on their application development without wanting to care about their infrastructure and they also want that they pay for what they use. Hence, AWS Lambda comes into play which satisfies all their needs.

Ironically, Lambda could be a threat to one of the Amazon's most popular EC2. Developers can build apps that run entirely on Lambda functions instead of spinning up EC2 VMs. Amazon may be out-innovating itself with Lambda.

In AWS Lambda, we have triggers. Lambda Functions can be triggered in different ways: an HTTP request, a new document upload to S3, a scheduled Job, an AWS Kinesis data stream, or a notification from AWS Simple Notification Service (SNS).

#### 14.2.6 AWS Lambda Example

Let us create our first Lambda function.

Step 1: The very first thing we need is an AWS account. (There is already a section on this, please go through that to understand how to create an AWS account - [Creating AWS account](#)

Step 2: We will be writing a function that we call `isPalindrome`, which will check if the string is palindrome or not.

```
function isPalindrome(string) {
 const reverse = string.split('').reverse().join('');
 const isPalindrome = (string === reverse);
 const result = isPalindrome ? `${string} is a Palindrome` : `${string} is not a Palindrome`;
 return result;
}

document.write(isPalindrome('abcd'));
```

This example we store in a file as javascript named `isPalindrome.js`

Step 3: Let's see how to create an AWS Lambda function - `isPalindrome`. Firstly, go to AWS Console. (see [Figure 108](#)).

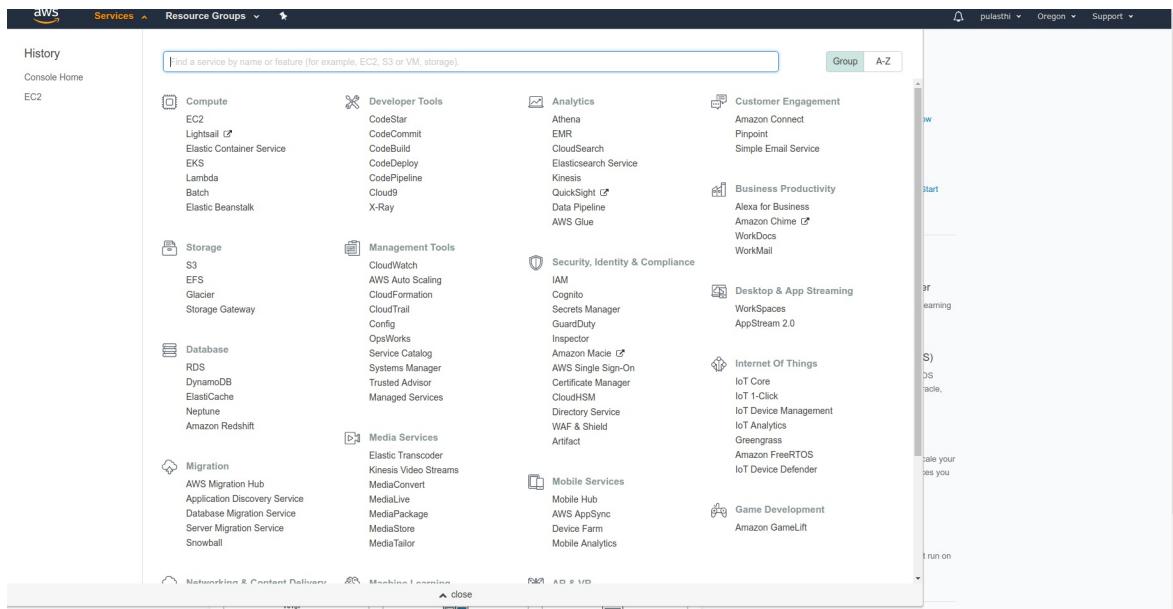


Figure 108: AWS Console

Step 4: Now we will select AWS Lambda from console and then click on **Get Started Now** (see [Figure 109](#))

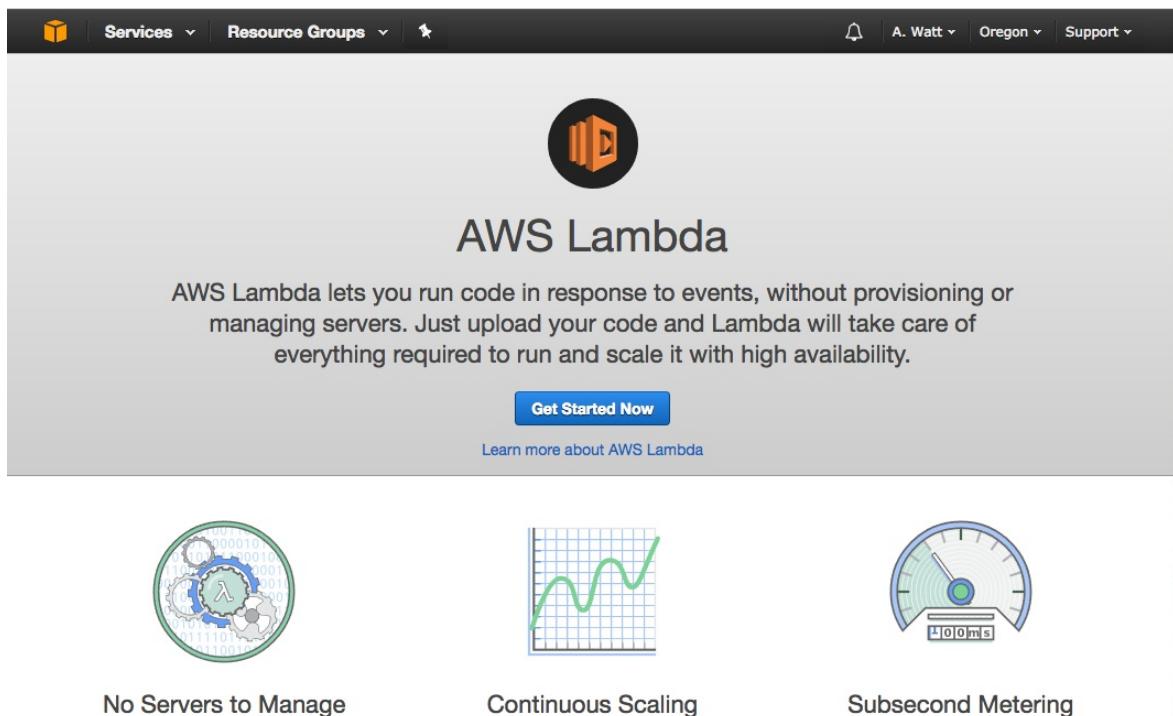


Figure 109: AWS Lambda

Step 5: For runtime, we will select Node.js 6.10 and then press "Blank Function." (see [Figure 110](#)).

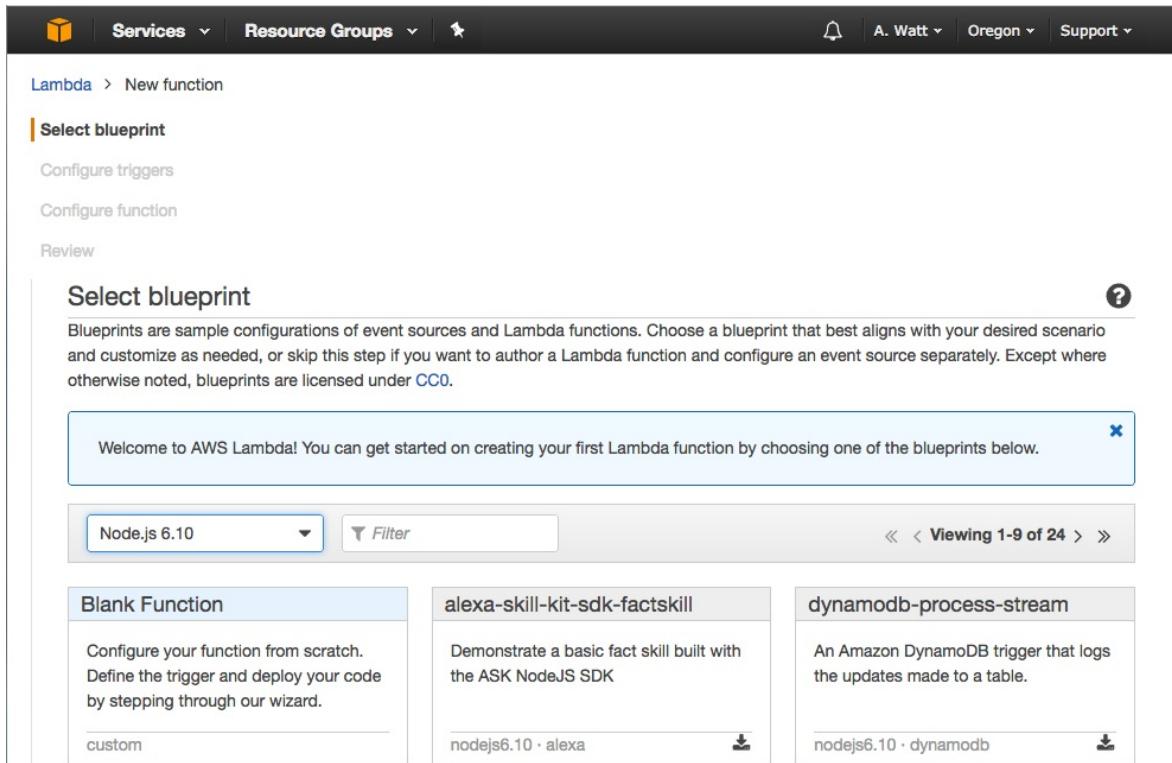


Figure 110: Blank Function

Step 6: We will skip this step and press **Next**. (see [Figure 111](#))

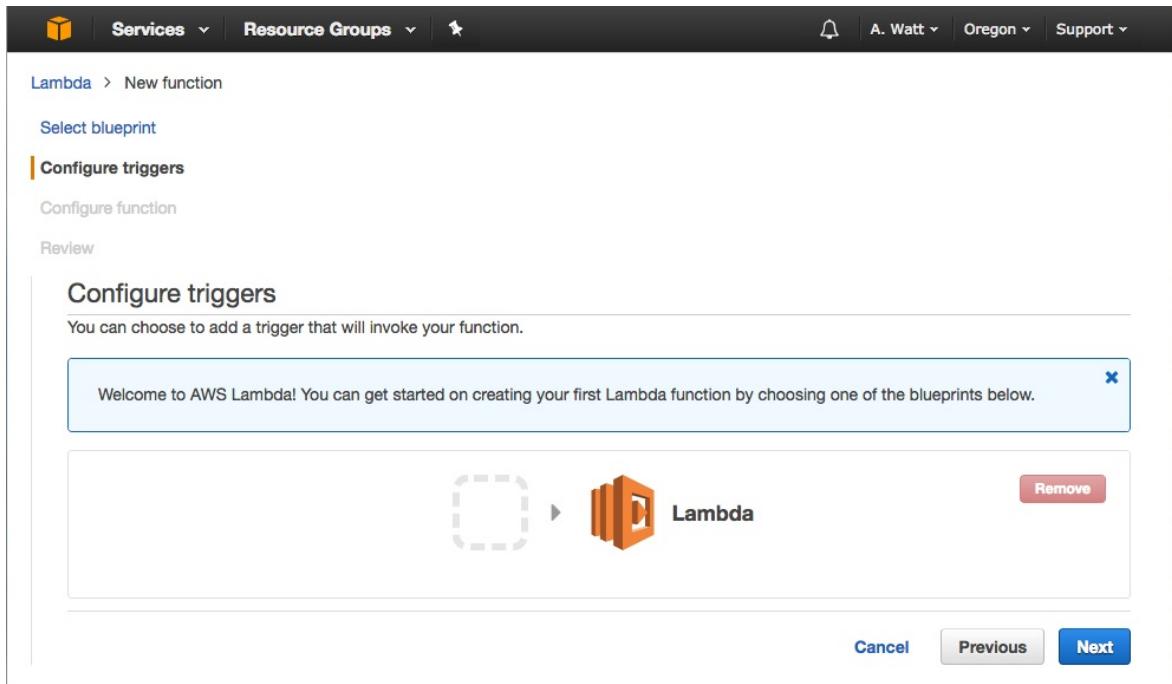


Figure 111: Next

Step 7: Let's give the Name as isPalindrome and put in a description of our new Lambda Function, or we can leave it blank. (see [Figure 112](#))

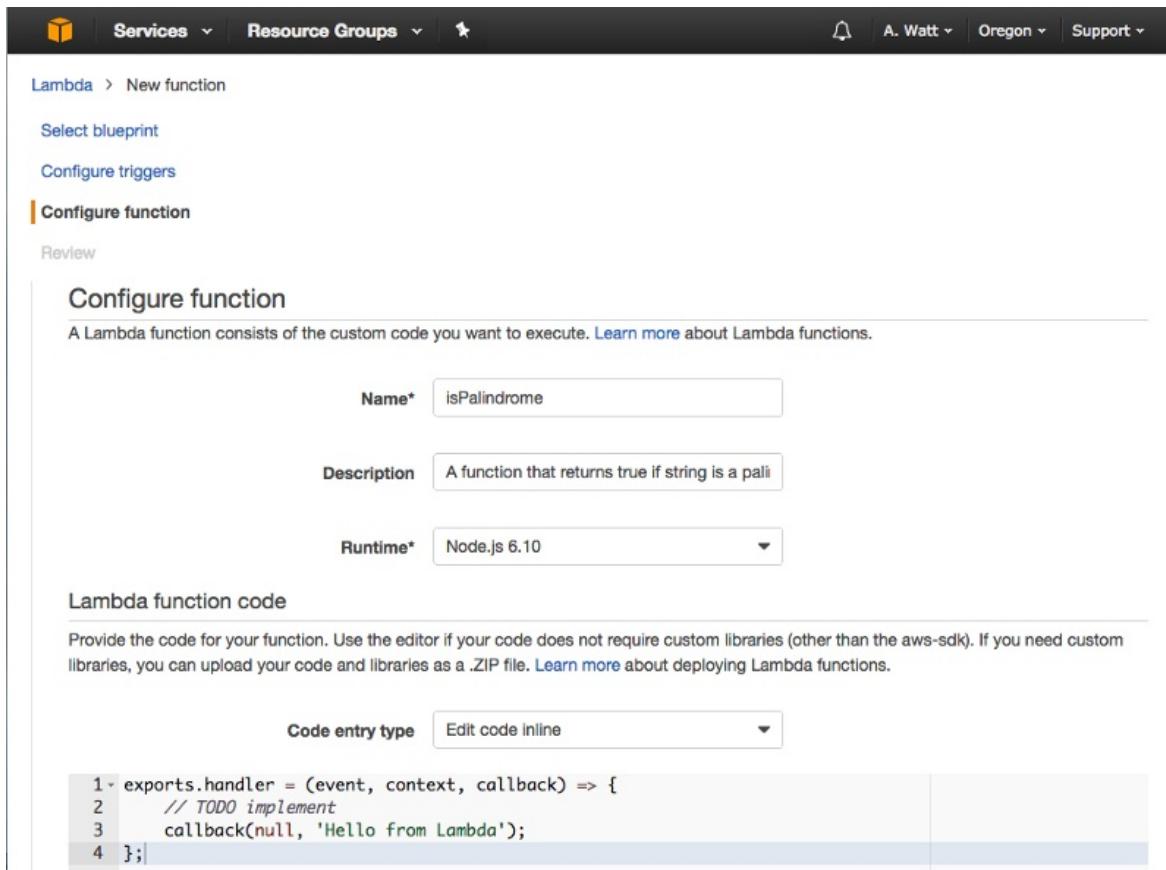


Figure 112: Description

Lambda function is just a function, named as handler here and the function takes three parameter - event, context and a callback function. The callback will run when the Lambda function is done and will return a response or an error message. For the Blank Lambda blueprint, response is hard-coded as the string `Hello from Lambda`.

Step 8: Please scroll down for choosing the Role “Create new Role from template”, and for Role name we are going to use `isPalindromeRole` in our case. For Policy templates, we will choose “Simple Microservice” permissions. (see [Figure 113](#))

You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#). For storing sensitive information, we recommend encrypting values using KMS and the console's encryption helpers.

Enable encryption helpers

Environment variables	Key	Value	X
-----------------------	-----	-------	---

Lambda function handler and role

Handler\*  ⓘ

Role\*  ⓘ

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name\*  ⓘ

Policy templates  ⓘ

Tags

Advanced settings

\* These fields are required.

Cancel Previous Next

Figure 113: Policy

Step 9: For Memory, 128 megabytes is more than enough for our simple function. As for the 3 second timeout, this means that—should the function not return within 3 seconds- AWS will shut it down and return an error. Three seconds is also more than enough. Leave the rest of the advanced settings unchanged (see @#fig:aws-lambda-settings).

▼ Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)\*  ▼ ⓘ

Timeout\*  min  sec

---

AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's [retry policy](#) and [DLQs](#). Please ensure your role has appropriate permissions to access the DLQ resource.

DLQ Resource  ▼ ⓘ

---

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. [Learn more](#) about accessing VPCs within Lambda. Please ensure your role has appropriate permissions to configure VPC.

VPC  ▼ ⓘ

---

AWS X-Ray provides tracing and monitoring capabilities for your Lambda function. Click [here](#) to learn more.

Enable active tracing  ⓘ

---

Environment variables are encrypted at rest using a default Lambda service key. You can change the key below to one of your account's keys or paste in a full KMS key ARN.

KMS key  ▼ ⓘ

---

\* These fields are required.

Cancel Previous Next

Figure 114: Advanced Settings

Step 10: Let's click on the "Create function" button now to create our first Lambda function. (see [Figure 115](#))

**Review**

**Review**

Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.

**Lambda function**

**Name** isPalindrome **Edit**

**Description** A function that returns true if string is a palindrome

**Runtime** Node.js 6.10

**Environment variables**

**Handler** index.handler

**Role name\*** isPalindromeRole

**Policy templates** Simple Microservice permissions

**Tags**

**DLQ Resource**

**Memory (MB)** 128

**Timeout** 3

**VPC** No VPC

**Enable active tracing**

**KMS key** (default) aws/lambda

**Cancel** **Previous** **Export function** **Create function**

Figure 115: Create

Step 11: Now that we have created our first Lambda function, let's test it by clicking **Test** (see [Figure 116](#))

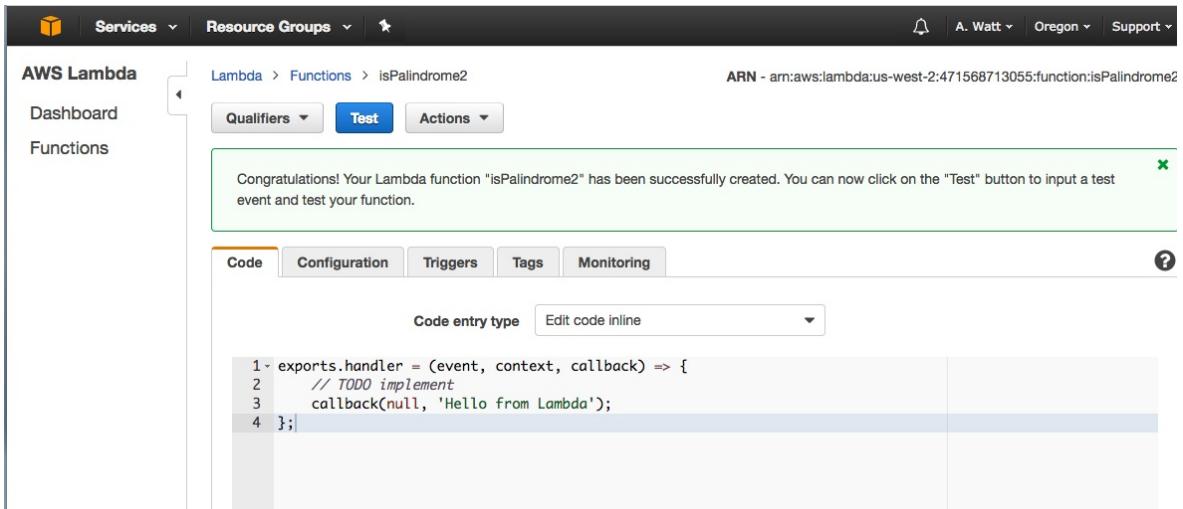


Figure 116: Test

The output will be the hard-coded response of `Hello from Lambda.` from the created Lambda function. (see [Figure 117](#))

Summary	Log output
<b>Code</b> rX2q+rcKE/bOXNhSIUo <b>SHA-256</b> nxOP6qv3FdnZr0UQi4g <b>ZKUGA=</b>  <b>Request</b> ab9ca031-3cf2-11e7- <b>ID</b> b746-51fbc2f25f4b  <b>Duration</b> 0.41 ms  <b>Billed</b> 100 ms <b>duration</b>  <b>Resources</b> 128 MB <b>configured</b>  <b>Max</b> 18 MB <b>memory</b> <b>used</b>	<pre> START RequestId: ab9ca031-3cf2-11e7-b746-51fbc2f25f4b Version: \$LATEST END RequestId: ab9ca031-3cf2-11e7-b746-51fbc2f25f4b REPORT RequestId: ab9ca031-3cf2-11e7-b746-51fbc2f25f4b Duration: 0 </pre>

Figure 117: Hello

Step 12: Now let us add our `isPalindrome.js` function code here to

Lambda function but instead of return result use `callback(null, result)`. Then add a hard-coded string value of abcd on line 3 and press `Test`. (see [Figure 118](#))

```

AWS Lambda
Lambda > Functions > isPalindrome
ARN - arn:aws:lambda:us-west-2:471568713055:function:isPalindrome

Dashboard Functions
Qualifiers Test Actions
Code Configuration Triggers Tags Monitoring

Code entry type Edit code inline

1 exports.handler = (event, context, callback) => {
2
3 const string = 'abcd';
4 const reverse = string.split('').reverse().join('');
5 const isPalindrome = (string === reverse);
6
7 const result = isPalindrome ? `${string} is a Palindrome` : `${string} is not a Palindrome`;
8
9 callback(null, result);
10
11 };
12

```

Figure 118: Press Test

The output will be `abcd is not a Palindrome` (see [Figure 119](#))

Summary		Log output
<b>Code SHA-256</b>	FgmoNX64XarsbHE5DFpOoW/EtxnUE2IBGvW1+elZ/Wc =	The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. <a href="#">Click here</a> to view the CloudWatch log group.
<b>Request ID</b>	a640158f-3f15-11e7-835c-6529d02c41ba	START RequestId: a640158f-3f15-11e7-835c-6529d02c41ba Version: \$LATEST END RequestId: a640158f-3f15-11e7-835c-6529d02c41ba REPORT RequestId: a640158f-3f15-11e7-835c-6529d02c41ba Duration: 7.81 ms
<b>Duration</b>	7.81 ms	
<b>Billed</b>	100 ms	

Figure 119: Output

Similarly, let us try with string `abcdcba` and in this case output should return `abcdcba is a Palindrome`. Thus, our Lambda function is behaving as expected.

## 14.3 APACHE OPENWHISK



- this section includes many references to other tools, that need bibtex references.

Apache OpenWhisk is a Function as a Service (FaaS), aka Serverless computing, platform used to execute code in response of an events via triggers by managing the infrastructure, servers and scaling. The advantage of OpenWhisk over traditional long-running VM or container approach is that there is lack of resiliency-related overhead in OpenWhisk. OpenWhisk is inherently scalable since the actions are executed on demand. OpenWhisk also helps the developers to focus only on coding by taking care of infrastructure-related tasks like monitoring and patching.

The developers provide the code written in the desired programming language for the desired action and this code will be executed in response to the events. The triggering can be invoked using HTTP requests or external feeds. The events invoking the triggers ranges from database modification to new variables in IoT sensors. Actions that response to these events could also range from a Python code snippet to a binary code in a container and it is as well possible to chain the actions. Note that these actions are deployed and executed instantaneously and can be invoked not only by triggers but also using the OpenWhisk API or CLI.

### **14.3.1 OpenWhisk Workflow**

OpenWhisk uses Nginx, Kafka, Docker and CouchDB as internal components. To understand the role of each of these components, let's review an action invocation trace in the system. Remember the main outcome of OpenWhisk (or Serverless architecture in general) is to execute the user's code inside the system and return the result. The workflow of the OpenWhisk is illustrated in the figure [Figure 120](#)

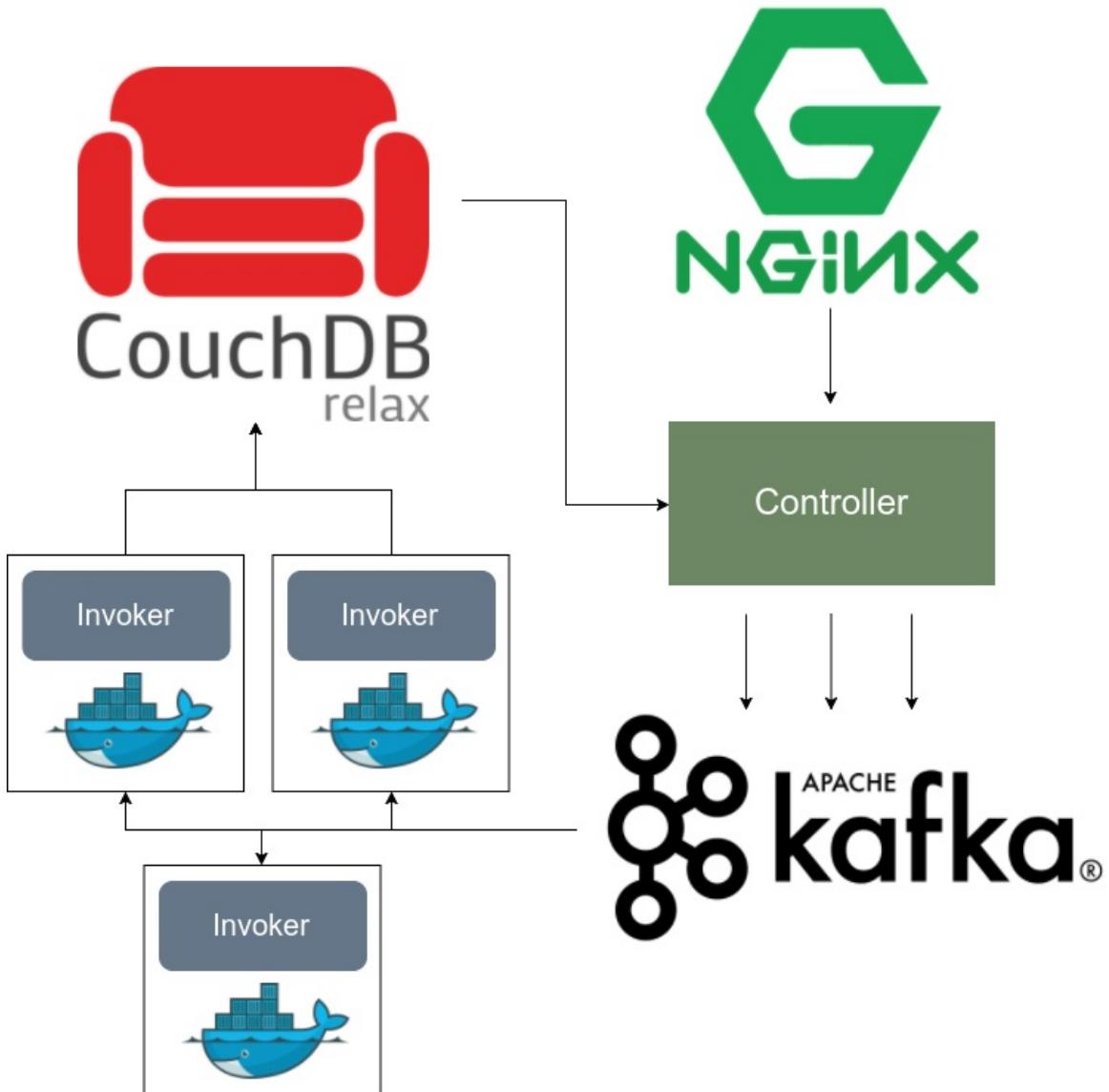


Figure 120: OpenWhisk workFlow

We will review the role of each components in the OpenWhisk workflow.

#### 14.3.1.1 The Action and Nginx

As mentioned prior, the action is the response of the OpenWhisk to triggers. Consider the following JavaScript function:

```
function main() {
 return { hello: 'world' };
```

```
}
```

This is the Hello World example of the OpenWhisk action where the action returns a JSON object with the key `hello` which has a value of `world`. After saving this function in a `.js` file, e.g. `action.js` then the action could be created using the following command:

```
$ wsk action create HelloAction action.js
```

Then, the `HelloAction` can be invoked using:

```
$ wsk action invoke HelloAction --result
```

The `wsk` command is what is known as OpenWhisk CLI, which we will show how to install in the next sections. Note that OpenWhisk's API is RESTful and fully HTTP based. In other words, the above-mentioned `wsk action` command is basically a HTTP request equivalent to the following:

```
POST /api/v1/namespaces/$userNamespace/actions/HelloAction
Host: $openwhiskEndpoint
```

The `userNamespace` variable defines the namespace in which the `HelloAction` is put into. Accordingly, nginx is the entering point of the OpenWhisk system and it plays an important role as a HTTP server as well as a reverse proxy server, mainly used for SSL termination and HTTP request forwarding.

#### 14.3.1.2 Controller: The System's Interface

We learned that nginx does not do any processing on the HTTP request except decrypting it (SSL Termination). The main processing of the request starts in the Controller. The controller plays the role of the interface for user both for actions and Create, Read, Update, and Delete (CRUD) requests, translating the user's POST request to action invocation. The controller has an essential role in OpenWhisk workflow and its role is not finished here and is partially involved in next steps as well.

#### 14.3.1.3 CouchDB

Naturally some notion of authentication is essentially required for the system. This authentication is performed by the Controller via CouchDB. The CouchDB instance has a specific database, namely `subjects` which contains the credentials and corresponding privileges. The credentials that corresponds to a request are verified against the `subjects` database and if the user's privileges satisfies the permissions required for the requested `HelloAction`, the action will be invoked. In our example, we are assuming that the `HelloAction` is in a namespace owned by the user, meaning that the user has the required permission to invoke the action.

After authentication and authorization using the `subjects` database, the record for the action `HelloAction` is load from `whisks` database. This record contains the code, the parameters consist of default parameters merged with user parameters, as well as the resource limits, e.g. maximum memory. The `HelloAction` record in `whisk` contains its code (listed above) and no parameters as the code does not get any parameters.

#### **14.3.1.4 Load Balancer**

Next comes the load balancer which is technically part of the controller and it is load balancer's responsibility to check the health status of the executors, known as `Invokers`, continuously. Load balancer is aware of the available invokers and select them for the actions accordingly.

#### **14.3.1.5 Kafka**

For a request user sends, there are two scenarios where things can go bad:

- Invocation is lost due to a crash
- Invocation has to wait for invokers to be available

Both of this scenarios can be handled with Kafka distributed messaging system. The action invocation mechanism with Kafka is as

follows:

The controller “publishes” a message to Kafka. This message contains the required action and corresponding parameters and is addressed to an Invoker chosen by the controller. Kafka responds to the HTTP request of the user with an `ActivationId` which could be used later by the user to get the result. OpenWhisk supports both synchronous and asynchronous invocation models. In the former model, the user’s HTTP request is terminated as the system accepts it. The latter model, known as blocking invocation, is otherwise.

#### 14.3.1.6 Invoker

As the heart of the OpenWhisk, the Invoker’s responsibility is to invoke the action. Invoker is implemented in Scala but it uses Docker for a safe and isolated execution. For each invoked actions, a container is spawned and the code as well as the parameters are passed to it. As soon as the result is obtained, the container is terminated.

The `Action` example is a node.js action and therefore the invoker will start a node.js container, inject our above-mentioned code to it, runs the code and gets the results, save the logs and terminates the node.js container.

#### 14.3.1.7 CouchDB again

The result of the Invoker is saved in another database in CouchDB, namely `activations`, under same `ActivationId` that was sent back to the user. The result of the `HelloAction` example containing the log in JSON format, would look like this:

```
{
 "activationId": "31809ddca6f64cf9de2937ebd44fbb9",
 "response": {
 "statusCode": 0,
 "result": {
 "hello": "world"
 }
 },
```

```
"end": 1474459415621,
"logs": [
 "2016-09-21T12:03:35.619Z stdout: Hello World"
,
"start": 1474459415595,
}
```

Similar to the same API call used for submitting the action, we can use OpenWhisk's API to retrieve the result using the ActivationId:

```
wsk activation get 31809ddca6f64cf9de2937ebd44fb9
```

### 14.3.2 Setting Up OpenWhisk Locally

There are several approaches to starting the OpenWhisk platform:

- Directly running the service
- Running using Kubernetes and Mesos
- Running with Vagrant using a pre-configured VM

But an easier approach is using [OpenWhisk Devtools](#) which is purposed for local development and testing of OpenWhisk. Using OpenWhisk Devtools, you can quickly start OpenWhisk on any machine using `docker compose`. Accordingly, make sure the `docker compose` is already installed on your machine. Then to start the platform, clone the OpenWhisk Devtools and navigate to its folder, then:

```
$ cd docker-compose
$ make quick-start
```

Make sure you do not have any services running on the following ports otherwise the `docker compose` will fail starting some of the containers:

- 5984 for CouchDB
- 2181, 2888, 3888 for Zookeeper
- 9092 for Kafka
- 8888, 2551 for the Controller
- 8085 for the Invoker
- 9001 for Minio
- 6379 for Redis
- 8080, 443, 9000, 9090 for apigateway

- 8001 Kafka-UI

In case you have services running on any of the ports above, you can either stop the local services that are using these ports or alternatively you can modify the `docker-compose.yml` and change the source port number in the port number mapping. The latter option is, however, more tricky because you have to make sure the change does not affect the communication between the containers. For instance if you have `Apache` service running on Port 80, then open `docker-compose.yml`, search for the keyword `80:` to find the port mapping with source port of 80: `bash ports: - "80:80"`

then change it to another port:

```
ports:
 - "8080:80"
```

After that, you should be able to run `make quick-start` successfully and then you can check the status of the running docker containers using:

```
$ docker ps --format "{{.ID}}: {{.Names}} {{.Image}}"
16e7746c4af1: wsk0_9_prewarm_nodejs6 openwhisk/nodejs6action:latest
dd3c4c2d4947: wsk0_8_prewarm_nodejs6 openwhisk/nodejs6action:latest
6233ae715cf7: openwhisk_apigateway_1 openwhisk/apigateway:latest
3ac0938aecdd: openwhisk_controller_1 openwhisk/controller
e1bb7272a3fa: openwhisk_kafka-topics-ui_1 landoop/kafka-topics-ui:0.9.3
6b2408474282: openwhisk_kafka-rest_1 confluentinc/cp-kafka-rest:3.3.1
9bab823a891b: openwhisk_invoker_1 openwhisk/invoker
98ebd5b4d605: openwhisk_kafka_1 wurstmeister/kafka:0.11.0.1
65a3b2a7914f: openwhisk_zookeeper_1 zookeeper:3.4
9b817a6d2c40: openwhisk_redis_1 redis:2.8
e733881d0004: openwhisk_db_1 apache/couchdb:2.1
6084aec44f03: openwhisk_minio_1 minio/minio:RELEASE.2018-07-13T00-09-07Z
```

Note that 12 containers should be up and running:

```
$ docker ps --format "{{.ID}}: {{.Names}} {{.Image}} | wc -l"
12
```

### 14.3.2.1 Debugging quick-start

It is always possible that something goes wrong in the process of deploying the 12 dockers. If the `make quick-start` process stuck at some point, the best way to find the issue is to use the `docker ps -a` command

to check which of the containers is causing the issue. Then you can try to fix the issue of that container separately. This fix could possibly happen in `docker-compose.yml` file. For instance, there was some issue with the `openwhisk/controller` docker at some point and it turns out the issue was the following line in the `docker-compose.yml`:

```
command: /bin/sh -c "exec /init.sh --id 0 >> /logs/controller-local_logs.log 2>&1"
```

this line is indicating that the following command should be run after the container is started:

```
$ /init.sh --id 0 >> /logs/controller-local_logs.log 2>&1
```

However, starting another instance of the docker image with this command outputted a `Permission Denied` error which could be fixed either by changing the logs folder permission in the docker image or container (followed by a commit) or saving the log file in another folder. In this case replacing that line with the following line would temporarily fix the issue:

```
command: /bin/sh -c "exec /init.sh --id 0 >> /home/owuser/controller-local_logs.log 2>&1"
```

### 14.3.3 Hello World in OpenWhisk

OpenWhisk provides a command line tool called [openwhisk-cli](#) which is used for controlling the platform. As part of the `make quick-start` command that we used above for starting the platform, the account credentials will automatically be written into the configuration of the CLI. You can either install the CLI directly from the repository or install it using `linuxbrew`. Alternatively, use the binary available in this path in OpenWhisk Devtools folder:

```
[PATH_TO_DEVTOOLS]/docker-compose/openwhisk-src/bin/wsk
```

Running the `wsk` without any command or flag, will print its help:

```
~/incubator-openwhisk-devtools/docker-compose/openwhisk-src/bin$./wsk
```



```

\ \ \ / \ \
__\ tm

```

Usage:  
wsk [command]

Available Commands:

action	work with actions
activation	work with activations
package	work with packages
rule	work with rules
trigger	work with triggers
sdk	work with the sdk
property	work with whisk properties
namespace	work with namespaces
list	list entities in the current namespace
api	work with APIs

For instance, you can get the host address using:

```
$ wsk property get | grep host
whisk API host 192.168.2.2
```

You can then re-invoke the built-in `Hello World` example using:

```
~/incubator-openwhisk-devtools/docker-compose$ make hello-world
creating the hello.js function ...
invoking the hello-world function ...
adding the function to whisk ...
ok: created action hello
invoking the function ...
invocation result: { "payload": "Hello, World!" }
{ "payload": "Hello, World!" }
creating an API from the hello function ...
ok: updated action hello
invoking: http://192.168.2.2:9090/api/23bc46b1-71f6-4ed5-8c54-816aa4f8c502/hello/world
"payload": "Hello, World!"
ok: APIs
Action Verb API Name URL
/guest/hello get /hello http://192.168.2.2:9090/api/23bc46b1-71f6-4ed5-8c54-816aa4f8c502/hello/world
deleting the API ...
ok: deleted API /hello
deleting the function ...
ok: deleted action hello
```

#### 14.3.4 Creating a custom action

We already invoked the built-in hello world action. Now, we try to build a new custom action. First create a file called `greeter.js`:

```
function main(input) {
 return {payload: 'Hello, ' + input.user.name + ' from ' + input.user.location + '!'};
```

```
}
```

Now we can create an action called `greeter` using the `greeter.js`:

```
$ wsk -i action create greeter greeter.js
ok: created action greeter
```

Note that the `-i` option is to prevent the following error:

```
$ wsk action create greeter greeter.js
error: Unable to create action 'summer': Put https://192.168.2.2/api/v1/namespaces/guest/actions/greeter
Run 'wsk --help' for usage.
```

Afterwards you can get the list of actions to make sure your desired action is created:

```
$ wsk -i action list
actions
/guest/greeter private nodejs:6
```

Afterwards, we can invoke the action by passing a `json` parameter including a name and location and receive the result:

```
$ wsk -i action invoke -r greeter -p user '{"name": "Vafa", "location": "Indiana"}'
{
 "payload": "Hello Vafa from Indiana!"
}
```

Now we can retrieve the list of activation records:

```
$ wsk activation list -i
activations
976a7d02dab7460eaa7d02dab7760e9a greeter
02e0e12118af43b0a0e12118afdf3b038 hello
10c2cddb0d2c4c1f82cddb0d2c1c1feb hello
```

The result of the command above is showing that the `hello` action has been invoked twice and the `greeter` action was invoked once. You can get more information about each of the activation using the `wsk -i activation get [ACTIVATION_ID]`:

```
$ wsk -i activation get 976a7d02dab7460eaa7d02dab7760e9a
ok: got activation 976a7d02dab7460eaa7d02dab7760e9a
{
 "namespace": "guest",
 "name": "greeter",
 "version": "0.0.1",
 "subject": "guest",
```

```

"activationId": "976a7d02dab7460eaa7d02dab7760e9a",
"start": 1539980284774,
"end": 1539980284886,
"duration": 112,
"response": {
 "status": "success",
 "statusCode": 0,
 "success": true,
 "result": {
 "payload": "Hello Vafa from Indiana!"
 }
},
...

```

Finally, after you are finished using the OpenWhisk Devtools, you can stop platform using:

```

~/incubator-openwhisk-devtools/docker-compose$ make destroy
Stopping openwhisk_apigateway_1 ... done
Stopping openwhisk_controller_1 ... done
Stopping openwhisk_kafka-topics-ui_1 ... done
Stopping openwhisk_kafka-rest_1 ... done
Stopping openwhisk_invoker_1 ... done
Stopping openwhisk_kafka_1 ... done
Stopping openwhisk_zookeeper_1 ... done
Stopping openwhisk_redis_1 ... done
Stopping openwhisk_db_1 ... done
Stopping openwhisk_minio_1 ... done
...

```

## 14.4 KUBELESS



○ add bibtex

### 14.4.1 Introduction

Kubeless is an Serverless or FaaS framework that has been developed as a Kubernetes native framework. Kubeless is designed using services and features that are provided natively on the Kubernetes framework. It uses Kubernetes Custom Resource Definition to define functions

### 14.4.2 Programming model

Similar to other serverless frameworks, the programming model of Kubeless is an event-driven model. The two main components that

need to be understood and functions and events. Kubeless currently supports 3 types of functions runtimes Python, NodeJS and Ruby. These runtimes can be used to create and deploy functions. For each function an event type is defined which specifies the type of trigger for the event. Kubless currently supports 3 types of triggers which are, HTTP based, scheduled and event-based (pubsub).

### **14.4.3 System Architecture**

The system architecture is based completely on Kubernetes primitives which were discussed to some extent in the previous section. The Kubless architecture has 3 main components, Functions API, Kubeless-controller, and Kafka. Additionally, they provide Kubeless command-line client which can be used to perform CRUD operations for function more easily.

The Functions API provides a REST Endpoint to create, read, update and delete functions. This is developed as a Kubernetes Custom Resource Definitions (CRD). CRD is an extension point provided by Kubernetes that can be used to create custom resources. A custom resource exposes a REST endpoint and makes it available as any other REST API that is embedded with Kubernetes. Once created, Functions custom resource exposes the REST API that can be used for function CRUD operations. The Kubeless-controller is a custom controller that is deployed with the Kubernetes installation. This controller continuously monitors invocations that occur at the functions REST API and performs the required tasks according to the invocation. For example, if the invocation is for a function creation, the controller will create a Deployment for the function and a Service to expose the function. The Deployment will contain information on what runtime the function is intended to use, therefore the deployment will make sure to spin up Pods which will host containers of that runtime when a function execution is requested. Kafka is deployed within the Kubernetes installation as an event source which can be used to trigger the functions.

Because the container image that is used to execute the function is

generic, it does not have any specific dependencies that are required by the function and the function code itself. These two need to be injected into the Pod when the Pod is created. For the application logic/function code, Kubless uses a configuration resource provided by Kubernetes API named ConfigMap. The code segment is attached to the ConfigMap which can be read from within a Pod once the Pod is created. In order to install all the required dependencies, another Kubernetes resource named Init containers are utilized. Init containers are a special kind of container which can be configured to run when a Pod is created. Kubernetes also guarantees that all init containers specified for a Pod will run till completion before the application containers (in this case function container) are executed. Kubless runs an init container which will install all the required dependencies for the function before invoking the function. The function dependencies must be specified at function creation time.

## **14.5 MICROSOFT AZURE FUNCTION** FA18-516-08

---



○ TODO students can contribute this section

## **14.6 GOOGLE CLOUD FUNCTIONS** FA18-516-08

---



### Learning Objectives

- Introduction to Google Cloud Function
  - Practical example using console mode
- 

Google Cloud Function is Google's offering for Function as a Service. It enables serverless computing and offers functions as services on Google Cloud Platform driven by trigger events from Cloud Pub/Sub, Cloud Storage, HTTP or changes in log in Stackdriver logging. Cloud functions can also be invoked for real time mobile changes. Google Cloud page on cloud functions

- <https://cloud.google.com/functions/use-cases/>

gives more detail about the use cases such as Serverless application backends, Real-time data processing, Intelligent applications - all without the need of provisioning a server instance or the overhead of managing a server instance. The functions are invoked as services whenever needed for a business requirement and the cost is billed as per the minutes of usage just for the function execution time. The Google Cloud Functions can be written in Node.js or Python. For Python runtime environment refer to the page

- <https://cloud.google.com/functions/docs/concepts/python-runtime>

For Node.js 6 runtime environment refer to the page

- <https://cloud.google.com/functions/docs/concepts/nodejs-6-runtime>

For Node.js 8 runtime environment refer to the page

- <https://cloud.google.com/functions/docs/concepts/nodejs-8-runtime>

### 14.6.1 Google Cloud Function Example

Following from the example as presented in AWS Lambda section, we will look into a simple example of building a Google Cloud Function to check if a string is Palindrome or not. The implementation will be in Python and we will use Python runtime environment. We will use HTTP trigger to invoke the function using HTTP request. We will also use the Google Cloud Console to build, deploy and test the function. Finally we will use HTTP url to send request to the function to get the result of our query.

Let us begin:

**Step 1:** Login to Google Cloud Platform with your GCP account. We are using free tier for this demonstration. Refer to the section for Google Cloud in the epub for creating a free tier GCP account.

**Step 2:** Select or create a Project and go to dashboard (see [Figure 121](#))

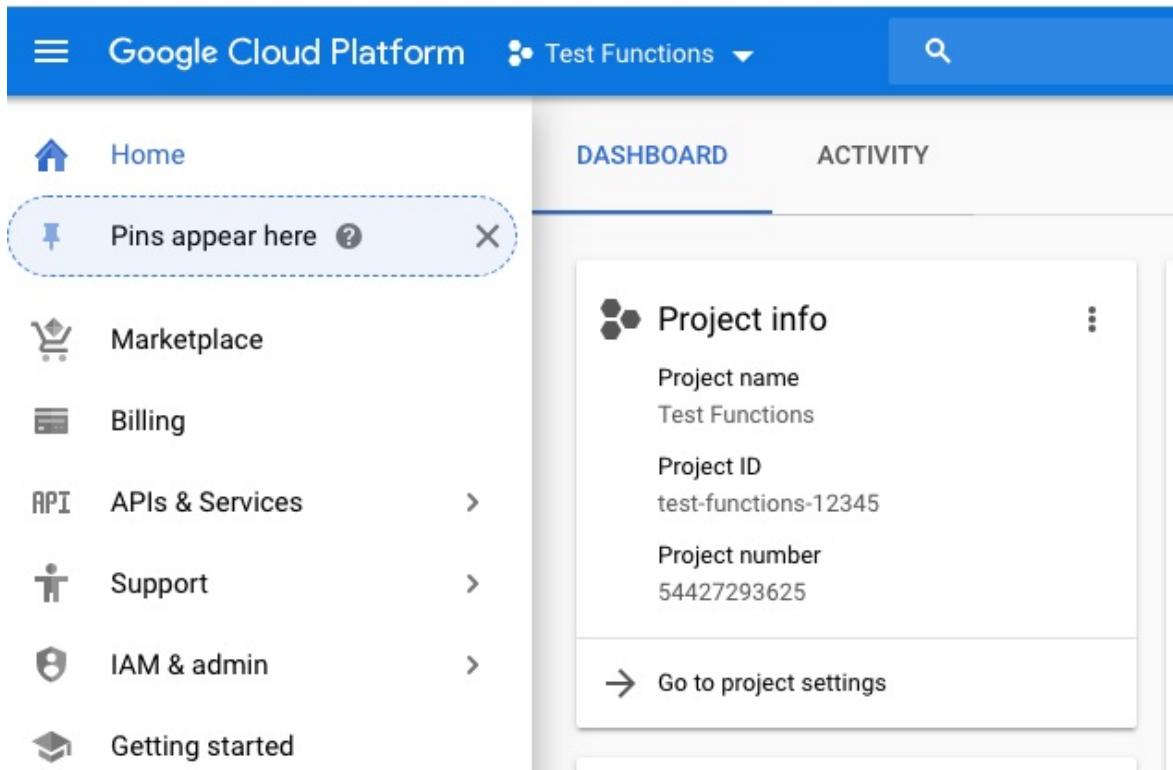


Figure 121: Login to Project and Dashboard

**Step 3:** Click “Create a Cloud Function” (see [Figure 122](#))

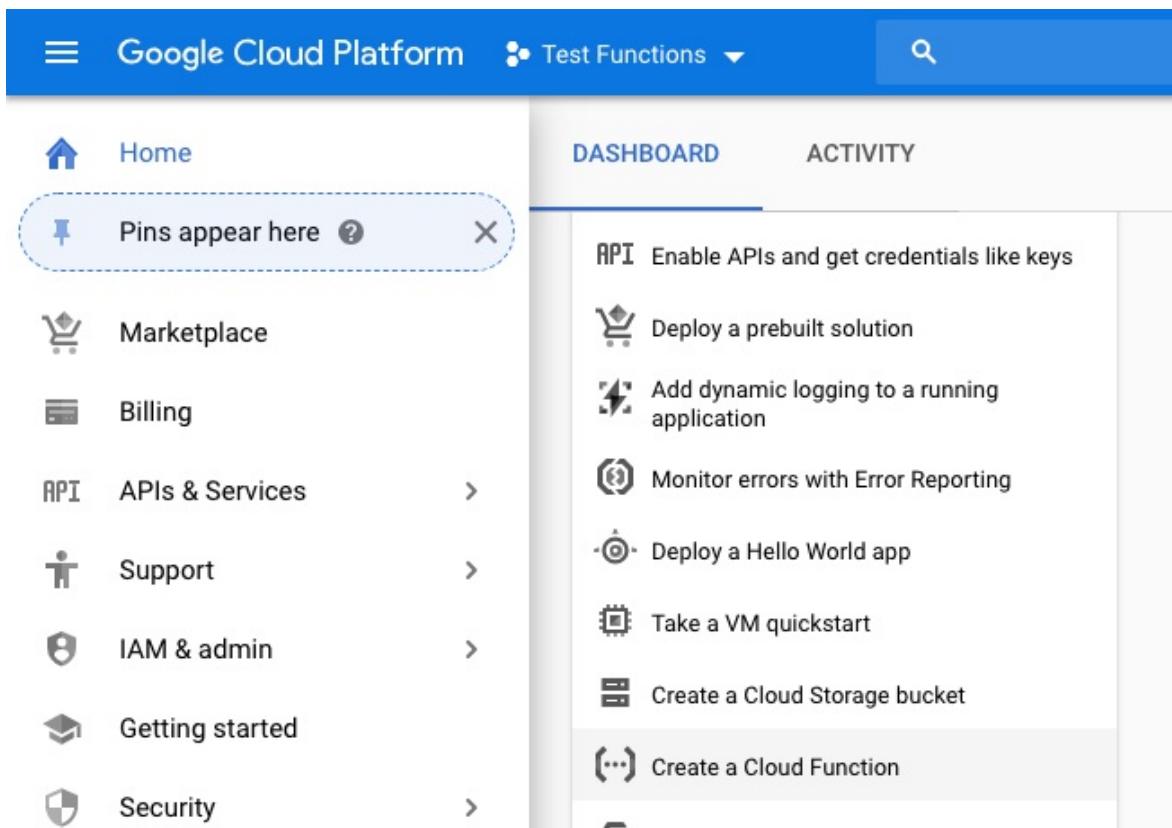


Figure 122: Create a Function

**Step 4:** Enable cloud function API if it is not enabled:(see [Figure 123](#))

This screenshot shows the "Cloud Functions" page in the Google Cloud Platform. At the top, it says "Google Cloud Functions". Below that, a message states: "Google Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment". Underneath this message, there is a yellow warning icon followed by the text "Cloud Functions API not enabled". At the bottom of the box, there is a blue button labeled "Enable API".

Figure 123: Enable the API

**Step 5:** Click Create Function (see [Figure 124](#))

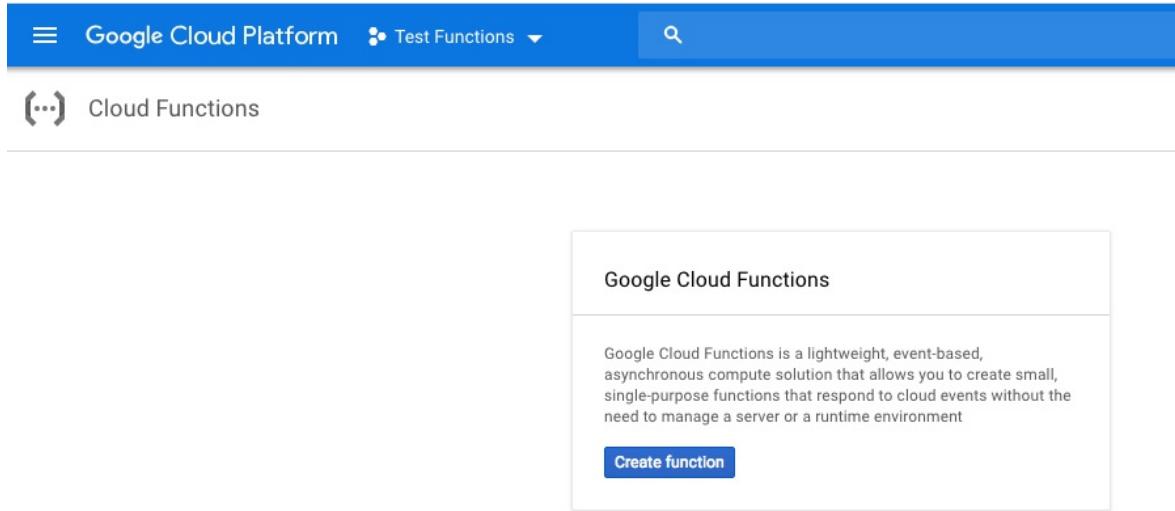


Figure 124: Select Create Function

**Step 6:** In the next page, give a name to the function. In our case we are giving function name as isPalindrome. Specify the memory (128 mb is good for this demo). Select the function trigger as HTTP. Choose inline editor for the source code and finally Python 3.7 as the run time environment.(see [Figure 125](#))

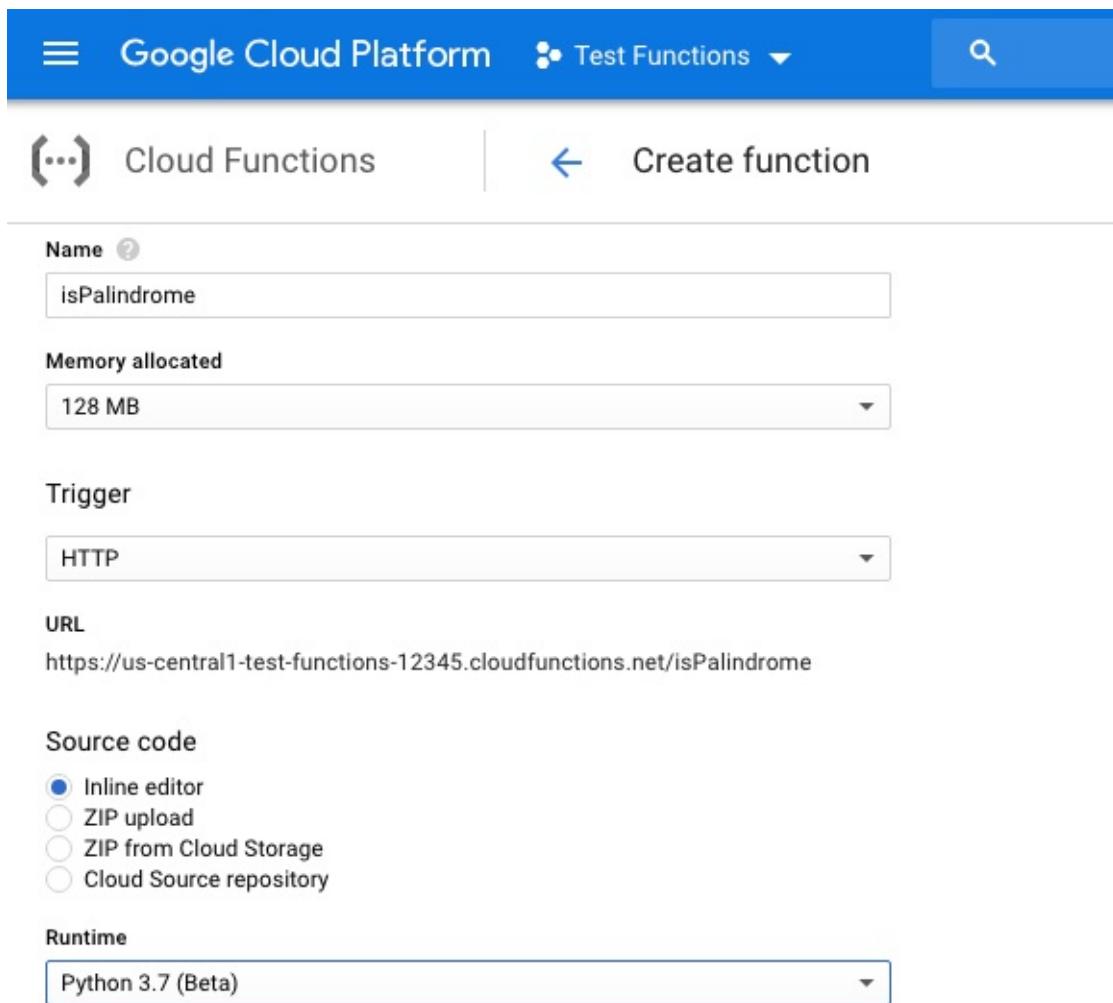


Figure 125: Name Function

**Step 7:** In the inline source editor, write a Python function and then click Create. We have written a Python function to check for Palindrome string. NOTE: This is not an optimized Python code, it is just used here for demonstration purpose. This function can be optimized further with Python standards style writing.(see [Figure 126](#))

The screenshot shows the Google Cloud Platform Cloud Functions interface. At the top, there's a navigation bar with the Google Cloud logo, 'Google Cloud Platform', and a 'Test Functions' dropdown. Below the navigation bar, the main title is 'Cloud Functions' with a 'Create function' button. A 'Runtime' dropdown is set to 'Python 3.7 (Beta)'. The code editor displays 'main.py' and 'requirements.txt'. The 'main.py' file contains the following Python code:

```
1 def isPalindrome(request):
2 """Responds to any HTTP request.
3 Args:
4 request (flask.Request): HTTP request object.
5 Returns:
6 The response text or any set of values that can be
7 Response object using
8 `make_response <http://flask.pocoo.org/docs/0.12/ag
9 """
10 request_json = request.get_json()
11 if request.args and 'message' in request.args:
12 string=request.args.get('message')
13 if(string==string[::-1]):
14 return f'The string is a palindrome'
15 else:
16 return f'The string isn not a palindrome'
17 elif request_json and 'message' in request_json:
18 string=request_json['message']
19 if(string==string[::-1]):
20 return f'The string is a palindrome'
21 else:
22 return f'The string isn not a palindrome'
23 else:
24 return f'Not Valid Request'
```

Function to execute [?](#)  
Figure 126: Write Python Function

**Step 8:** The function is created and deployed in the next page (see [Figure 127](#)) and [Figure 128](#))

Cloud Functions		Overview	<a href="#">CREATE FUNCTION</a>	<a href="#">REFRESH</a>	<a href="#">DELETE</a>	<a href="#">Columns</a>
<input type="text"/> Filter functions						
<input type="checkbox"/>	Name ^	Region	Trigger	Runtime	Memory allocated	Executed function
<input type="checkbox"/>	isPalindrome	us-central1	HTTP	Python 3.7 (Beta)	128 MB	isPalindrome

Figure 127: Function is Deployed

Cloud Functions		Overview	<a href="#">CREATE FUNCTION</a>	<a href="#">REFRESH</a>	<a href="#">DELETE</a>	<a href="#">Columns</a>
<input type="text"/> Filter functions						
<input type="checkbox"/>	Name ^	Region	Trigger	Runtime	Memory allocated	Executed function
<input checked="" type="checkbox"/>	isPalindrome	us-central1	HTTP	Python 3.7 (Beta)	128 MB	isPalindrome

Figure 128: Function is Deployed

**Step 9:** Finally we will test the function (see [Figure 129](#))

The screenshot shows the Google Cloud Platform Cloud Functions Overview page. At the top, there's a blue header bar with the Google Cloud Platform logo, the text 'Test Functions', and a search icon. Below the header, there are navigation links for 'Cloud Functions', 'Overview', 'CREATE FUNCTION', 'REFRESH', 'DELETE', and 'Columns'. A table lists the 'isPalindrome' function with the following details:

Name	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed
isPalindrome	us-central1	HTTP	Python 3.7 (Beta)	128 MB	isPalindrome	10/31/18, 2:31 AM

A context menu is open over the 'isPalindrome' row, listing options: 'Copy function', 'Test function', 'View logs', and 'Delete'. The 'Test function' option is highlighted.

Figure 129: Test The Function

**Step 10:** In the Trigger event box, write a HTTP message request in JSON format and click Test the Function (see [Figure 130](#))

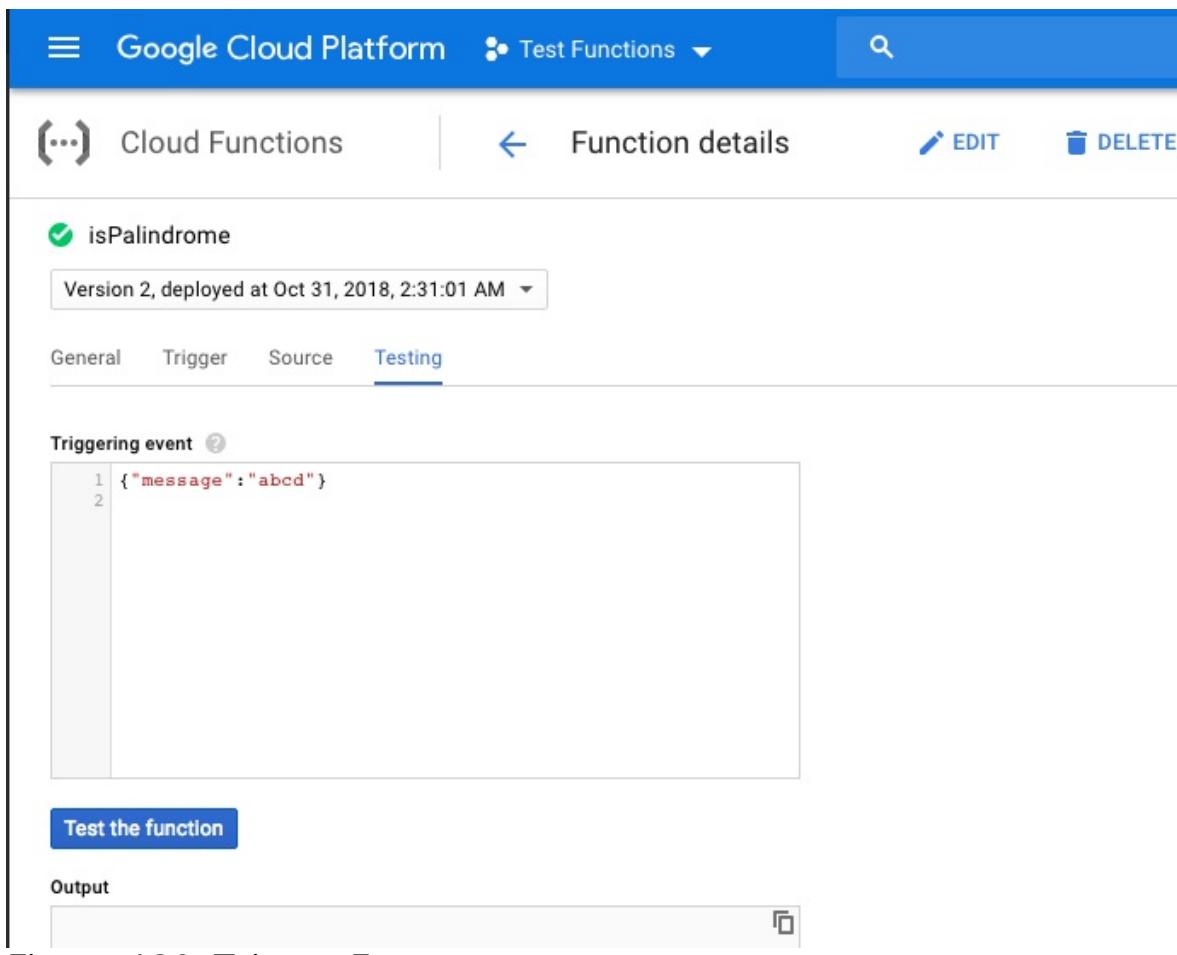


Figure 130: Trigger Event

**Step 11:** The response box will show the result of the test as expected (see [Figure 131](#))

The screenshot shows the Google Cloud Platform Cloud Functions interface. At the top, there's a blue header bar with the Google Cloud logo, the text "Google Cloud Platform", a "Test Functions" dropdown, and a search icon. Below the header, the main navigation bar has "Cloud Functions" on the left, a back arrow, "Function details" in the center, and an edit icon on the right. The function name "isPalindrome" is displayed with a green checkmark and a deployment timestamp: "Version 2, deployed at Oct 31, 2018, 2:31:01 AM". Below the function name are tabs: General, Trigger, Source, and Testing, with "Testing" being the active tab. Under "Triggering event", there's a code editor containing the following JSON:

```
1 {"message": "abcd"}
2
```

Below the code editor is a blue button labeled "Test the function". Under the "Output" section, the text "The string isn't a palindrome" is displayed next to a copy icon.

Figure 131: Result

**Step 12:** Let's run one more Test (see [Figure 132](#))

The screenshot shows the Google Cloud Platform Cloud Functions details page for a function named 'isPalindrome'. The top navigation bar includes the Google Cloud logo, 'Google Cloud Platform', 'Test Functions', and a search icon. Below the navigation, there's a breadcrumb trail with 'Cloud Functions' and a back arrow to 'Function details'. To the right is an edit icon with 'ED'. The main section displays the function name 'isPalindrome' with a green checkmark, and a deployment history entry: 'Version 2, deployed at Oct 31, 2018, 2:31:01 AM'. Below this are tabs for 'General', 'Trigger', 'Source', and 'Testing', with 'Testing' being the active tab. Under 'Triggering event', there's a code editor containing the following JSON payload:

```
1 {"message": "was saw"}
```

Below the code editor is a blue button labeled 'Test the function'.

Figure 132: Another Test

**Step 13:** You will get the expected result (see [Figure 133](#))

Triggering event ?

```
1 {"message": "was saw"}
2
```

**Test the function**

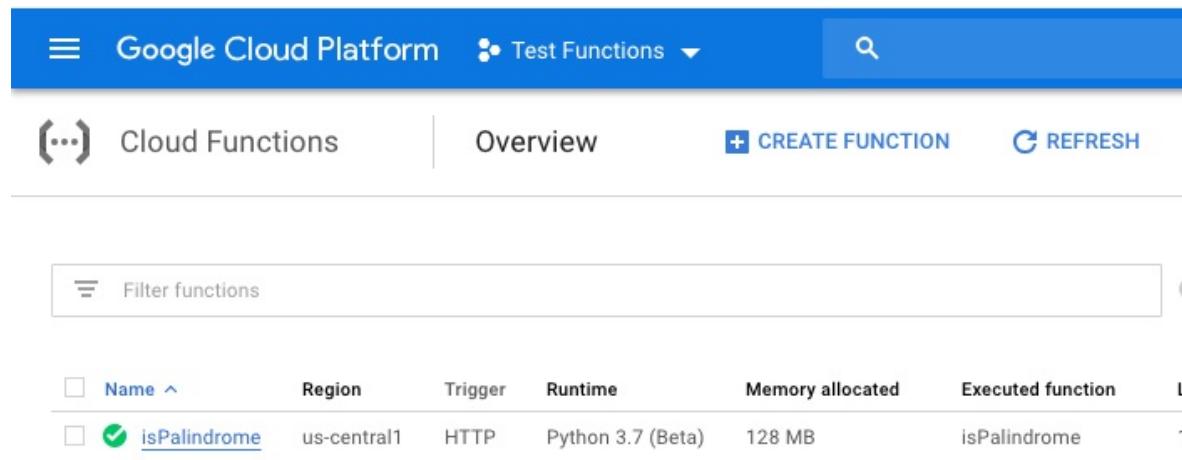
Output

```
The string is a palindrome
```



Figure 133: Expected Result For Test

**Step 14:** Let's test our function deployment using url. Click on the function name (see [Figure 134](#))



Google Cloud Platform Test Functions

Cloud Functions Overview CREATE FUNCTION REFRESH

Filter functions

Name	Region	Trigger	Runtime	Memory allocated	Executed function	L
<a href="#">IsPalindrome</a>	us-central1	HTTP	Python 3.7 (Beta)	128 MB	IsPalindrome	1

Figure 134: Deployment Url

**Step 15:** In the next page, click on Trigger page and copy the url (see [Figure 135](#))

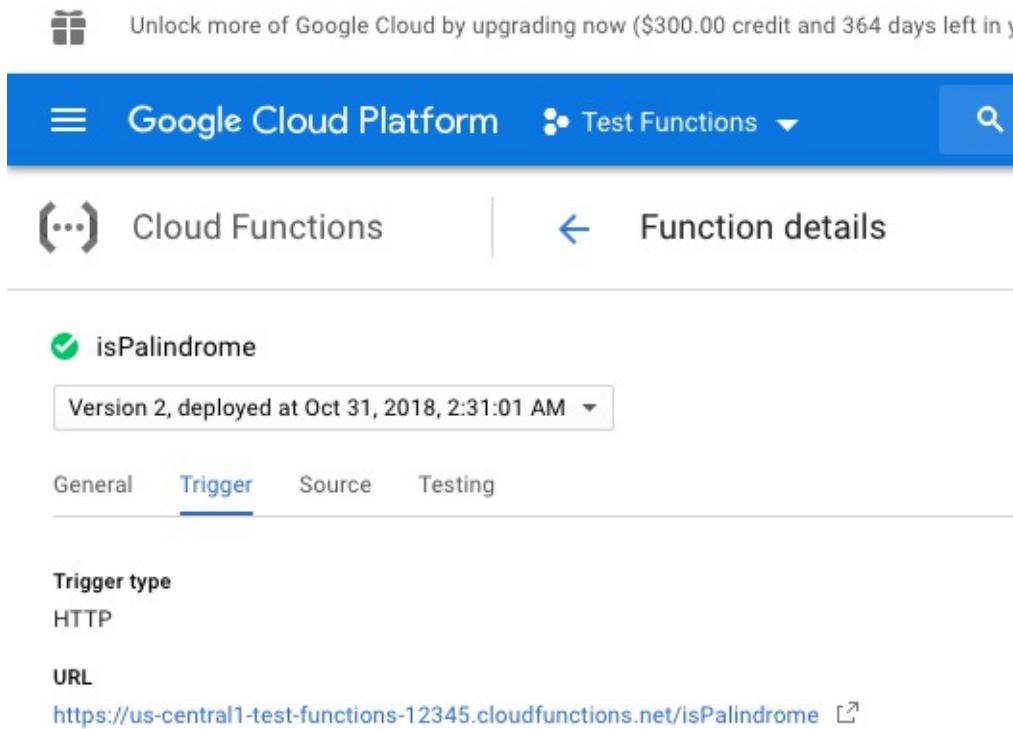


Figure 135: Trigger Url

**Step 16:** In a web browser type the url and add the HTTP request to it and hit enter

- <https://us-central1-test-functions-12345.cloudfunctions.net/isPalindrome?message=abcd>

**Step 17:** You will get a response back from the function(see [Figure 136](#))



Figure 136: Test Http

#### Step 18: Another test (see [Figure 137](#))



Figure 137: Another Http Test

This completes our demo for Google Cloud Function offered as Function as Service. To learn more about Google Cloud Functions and trigger options available alongwith triggers using command line - visit

<https://cloud.google.com/functions/>

To learn about creating and deploying functions using command line instead of GCP console - visit

- <https://cloud.google.com/functions/docs/quickstart>

## 14.7 OPENFAAS O 🖐 :FA18-516-23:



OpenFaas is a framework for building serverless functions on docker containers and follows the same workflow as micro services. Since, OpenFaas uses Docker and Kubernetes technologies, it will give lot of hosting options ranges from a laptop to large-scale cloud systems Any program written in any language can be packaged as a function

within in a container which gives a best approach to convert all the old code to run on cloud-based infrastructure

Few benefits of OpenFaas 1. Easy to Use 2. Deployable to private or public clouds in container 3. Simplicity in architecture and design 4. Open and extensible platform 5. Language agnostic

### 14.7.1 OpenFaas Components and Architecture

There are three components which includes API Gateway, Function Watchdog and the instance of Prometheus. All the functions are running on Docker containers orchestrated by either Docker Swarm or Kubernetes. The function watchdog is part of the function containers, whereas the API Gateway and Prometheus instance are services.

## Functions as a Service

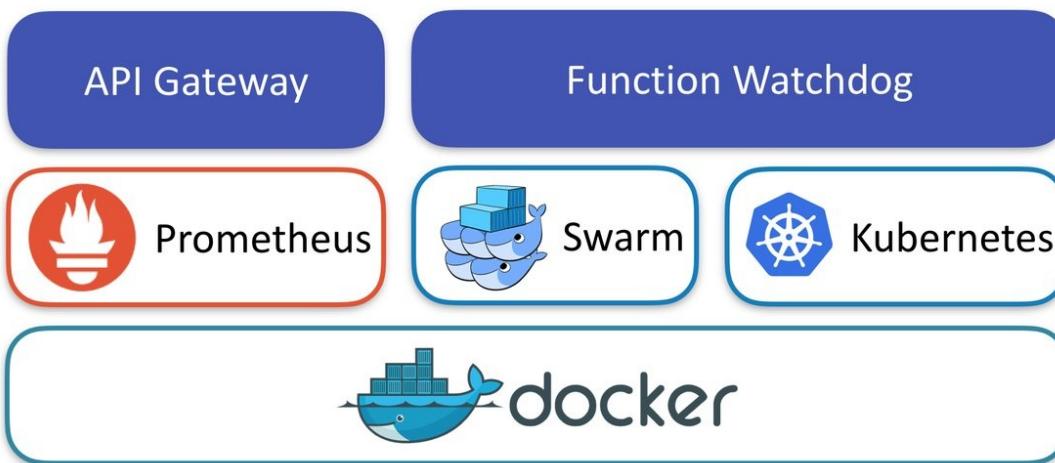


Figure 138: faas - OpenFaas - Arch [47]

#### 14.7.1.1 API Gateway

Routes inbound requests to the functions and collects metrics through Prometheus. It autoscales modifying service replicas counts.

Offers a convenient UI and endpoints for the CLI

#### **14.7.1.2 Function Watchdog**

It's a tiny HTTP server, enclosed along with the app in the docker image. It receives request from the API Gateway, triggers the app. It provide args and catch result through STDIN/STDOUT

#### **14.7.1.3 OpenFaas CLI**

The OpenFaas CLI provides mechanism to deploy the functions in the containders

#### **14.7.1.4 Monitoring**

OpenFaas makes monitoring simple with the use of Prometheus. The end users can install Grafana Dashboard and connect point to the Promotehus data source. This provides quick access to the dashboard to monitor the OpenFaas functions

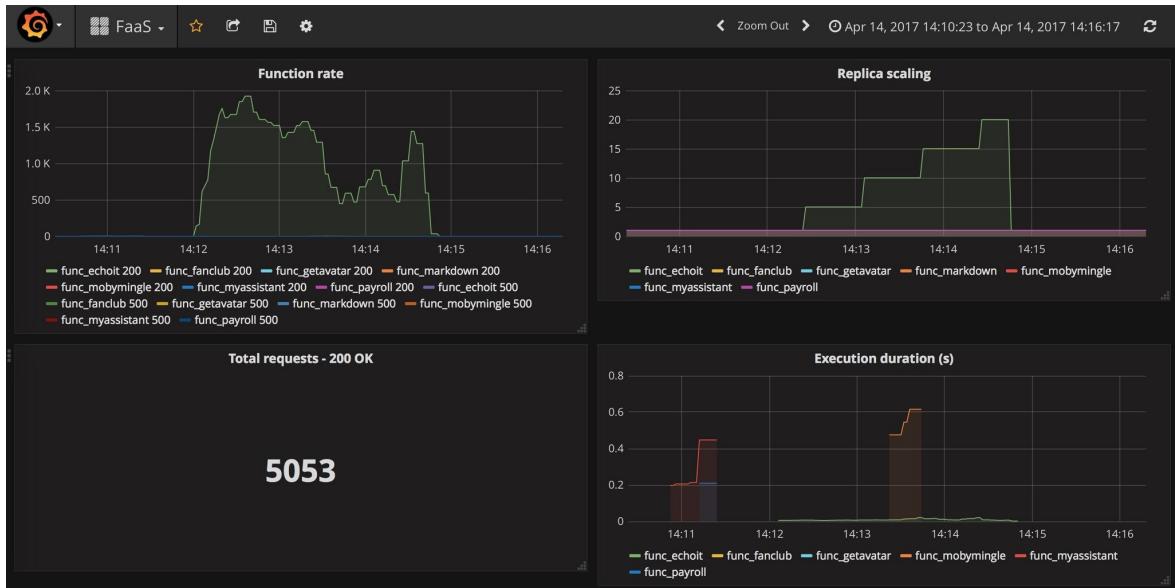


Figure 139: faas - OpenFaas - Grafana [47]

## 14.7.2 OpenFaas in Action

### 14.7.2.1 Prerequisites

1. Docker
2. Git Bash (for Windows)

### 14.7.2.2 Single Node Cluster

```
$ docker swarm init
```

Using a Terminal on Mac or Linux:

```
$ curl -sL cli.openfaas.com | sudo sh
```

For windows faas-cli.exe need to be downloaded from this link  
<https://github.com/openfaas/faas-cli/releases>

### 14.7.2.3 Deploy OpenFaas

OpenFaas gives the option to use yaml(.yml) file for configuring the functions and the image will be built by OpenFaas automatically.

Alternatively, custom docker image can be built and passed as an argument to the OpenFaas CLI. This gives the flexibility for the developers to extend further which is not in the standard yaml file.

```
$ git clone https://github.com/openfaas/faas
$ cd faas
$ git checkout master
$./deploy_stack.sh --no-auth
$ cd <test function folder>
$ docker build -t <test function image>
$ faas-cli deploy --image <test function image> --name <test function name>
```

#### 14.7.2.4 To Run OpenFaas

OpenFaas can be tested via curl, faas-cli, or any HTTP-based client to connect to the API gateway to invoke a function

Once the function is deployed, the functions can be verified in the following url <http://127.0.0.1:8080>

The screenshot shows the faas-OpenFaas-Portal interface. On the left, a sidebar lists deployed functions: func\_markdown, func\_echoit, func\_wordcount, func\_hubstats, and func\_base64. The main area displays the details for the 'func\_markdown' function. It shows 1 replica and 11 invocations. The image field is set to 'functions/markdown-render:latest@sha256:fbac7549ad7a598385c40df9e16d25c51a8786865'. Below this, there is an 'Invoke function' section with an 'INVOKE' button. Under 'Request body', the text '## The \*\*OpenFaas\*\* \_workshop\_' is entered. The response status is 200, and the round-trip time is 0.038 seconds. The response body contains the text '<h2>The <strong>OpenFaas</strong> <em>workshop</em></h2>'.

Figure 140: faas-OpenFaas-Portal [47]

## 14.8 RIFF O ?



○ TODO students can contribute this section

## **14.9 FISSION** ○ ?



○ TODO students can contribute this section

## **14.10 IRONFUNCTION** ○ ?



○ TODO students can contribute this section

## **14.11 FN** ○ ?



○ TODO students can contribute this section

## **14.12 GESTALT** ○ ?



○ TODO students can contribute this section

## **14.13 OPENLAMDA** ○ ?



○ TODO students can contribute this section

## **14.14 SPRING FUNCTION AS A SERVICE** ○ ?



○ TODO: student can contribute this section



## 15.1 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. These 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](#)

## 15.2 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 wil 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.

### **15.2.1 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 diskspace.

Please let us know the smallest configuration that works.

### **15.2.2 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 your 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 CONTROL v

Vbox to OSX:

shift CONTROL v shift CONTROL 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.

### 15.2.3 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.

## 15.3 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 supports 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>

### 15.3.1 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>

#### 15.3.1.1 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

#### 15.3.1.2 Windows ?

? students contribute

### 15.3.1.3 Linux O ?

? students contribute

### 15.3.2 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 to 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.

## 15.4 PACKER



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 from 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>

### 15.4.1 Installation

Installation instructions for all platforms is located at

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

### 15.4.2 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",
 "region": "us-central1",
 "zone": "us-central1-a",
 "disk_size": 10,
 "ssh_username": "root",
 "ssh_port": 22
 }
]
}
```

```

 "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/g1h4bhf54x750jzjyryckmn0
 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-ebs",
 "ssh_username": "{{ user `ssh_username` }}",
 "profile": "default",
 "ami_name": "{{ user `image_name` }}",
 "source_ami": "ami-0bbe6b35405ebebdb",
 "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's 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-ebs output will be in this color.
googlecompute output will be in this color.

==> googlecompute: Checking image does not exist...
==> amazon-ebs: 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-ebs: Found Image ID: ami-0bbe6b35405ecebdb
==> amazon-ebs: Creating temporary keypair:
 packer_5bad9d99-f631-1778-1e83-afdf19ad0d5cc
==> amazon-ebs: Creating temporary security group for this instance:
 packer_5bad9d9b-38c5-252d-0368-74aa75bfb286
==> amazon-ebs: Authorizing access to port 22 from 0.0.0.0/0
 in the temporary security group...
==> amazon-ebs: Launching a source AWS instance...
==> amazon-ebs: Adding tags to source instance
 amazon-ebs: Adding tag: "Name": "Packer Builder"
 amazon-ebs: 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.

## **15.5 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.

### **15.5.1 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 form 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 -1
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
```

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 than 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 computer.** 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.

### **15.5.1.1 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.

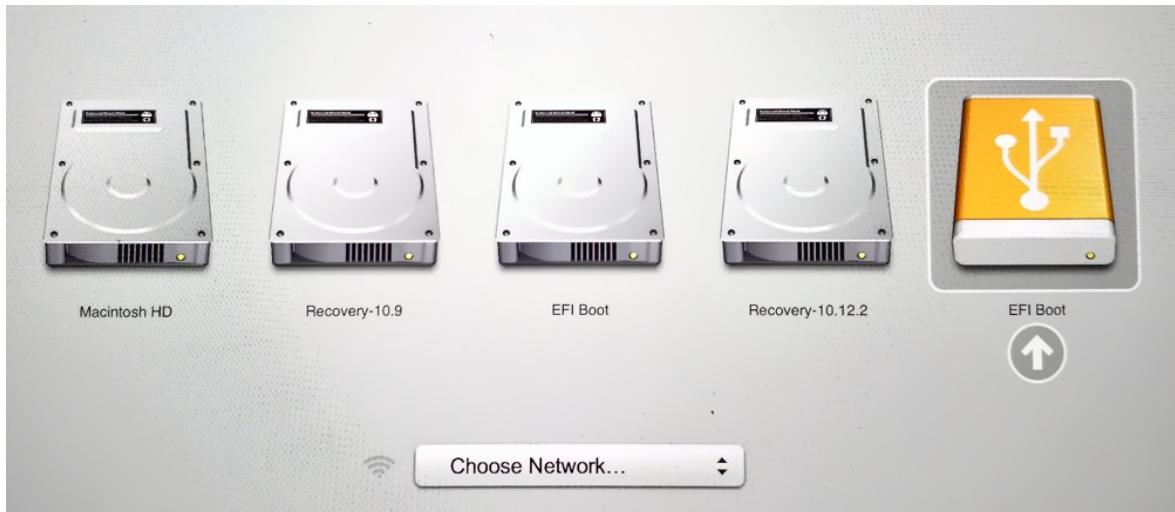


Figure: Boot Screen

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.

## **15.5.2 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.

### **15.5.2.1 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.

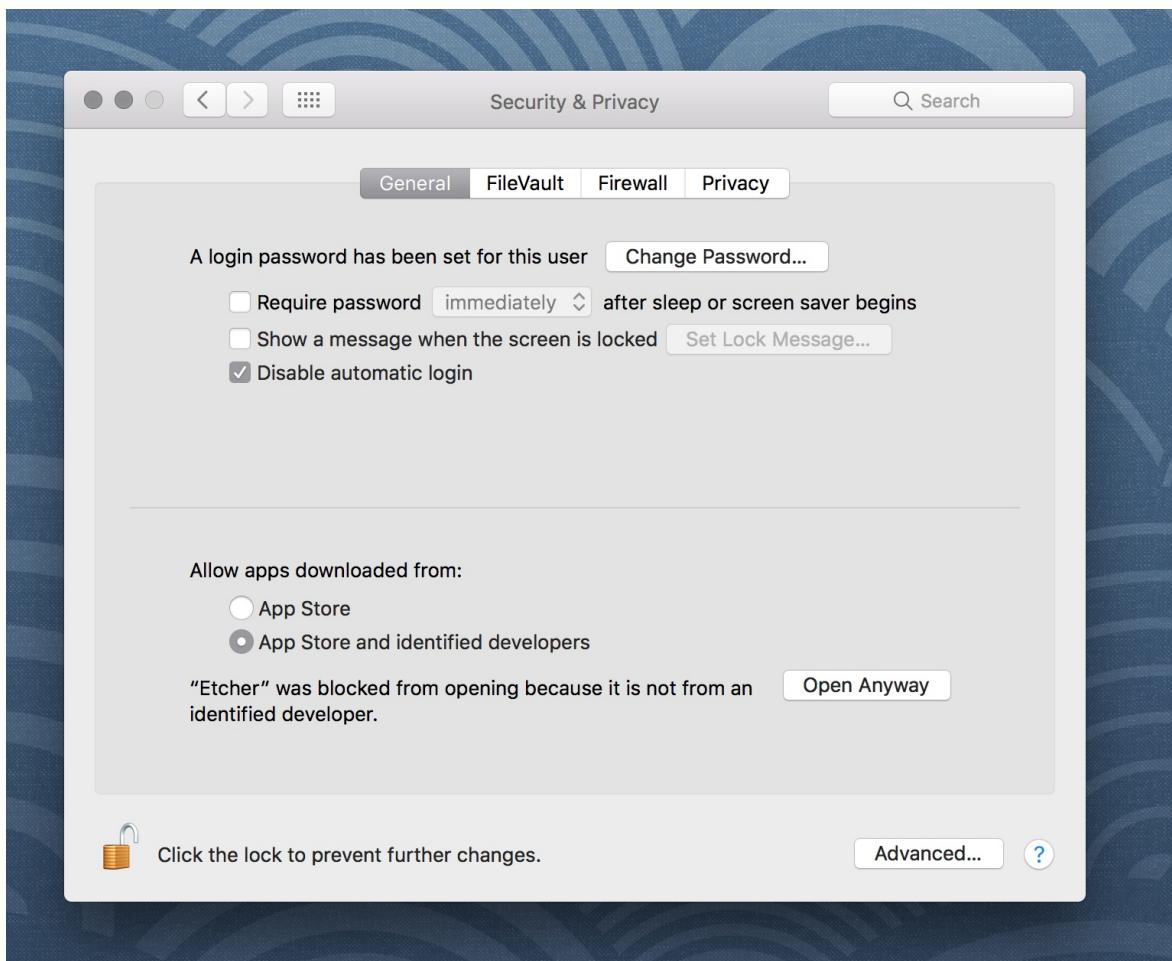


Figure: Setting

### 15.5.2.2 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

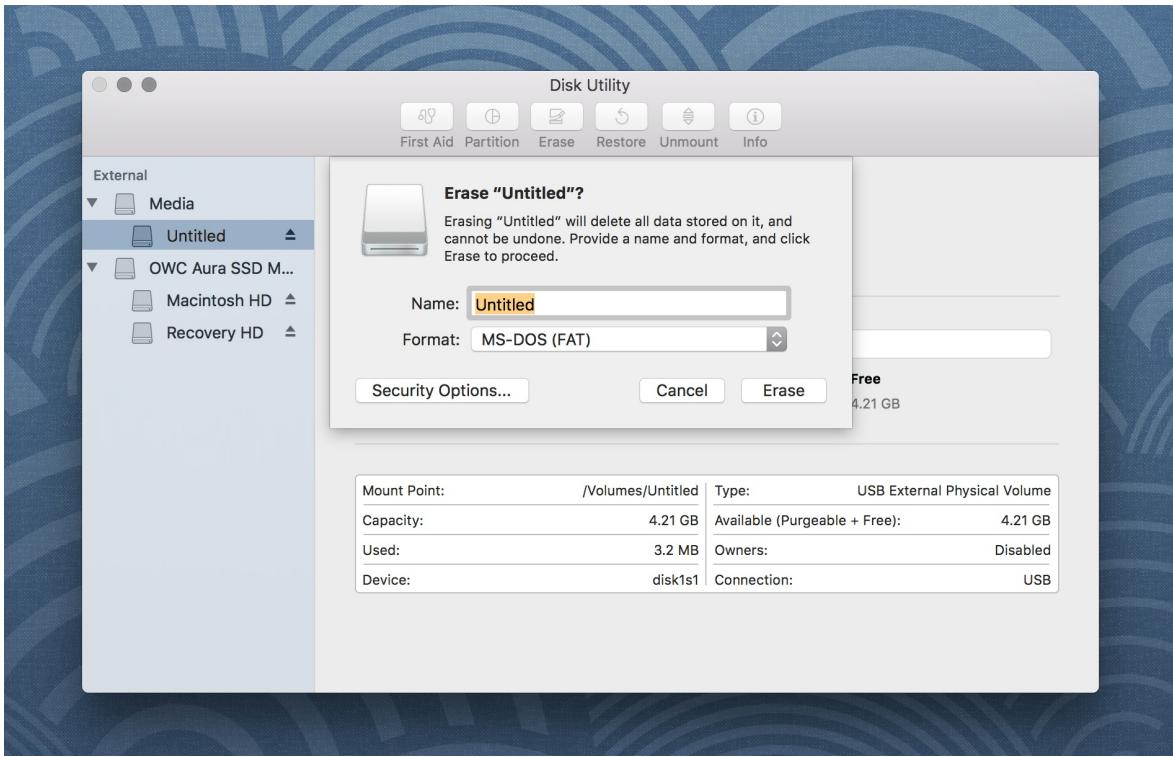


Figure: Diskutil

### 15.5.2.3 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.

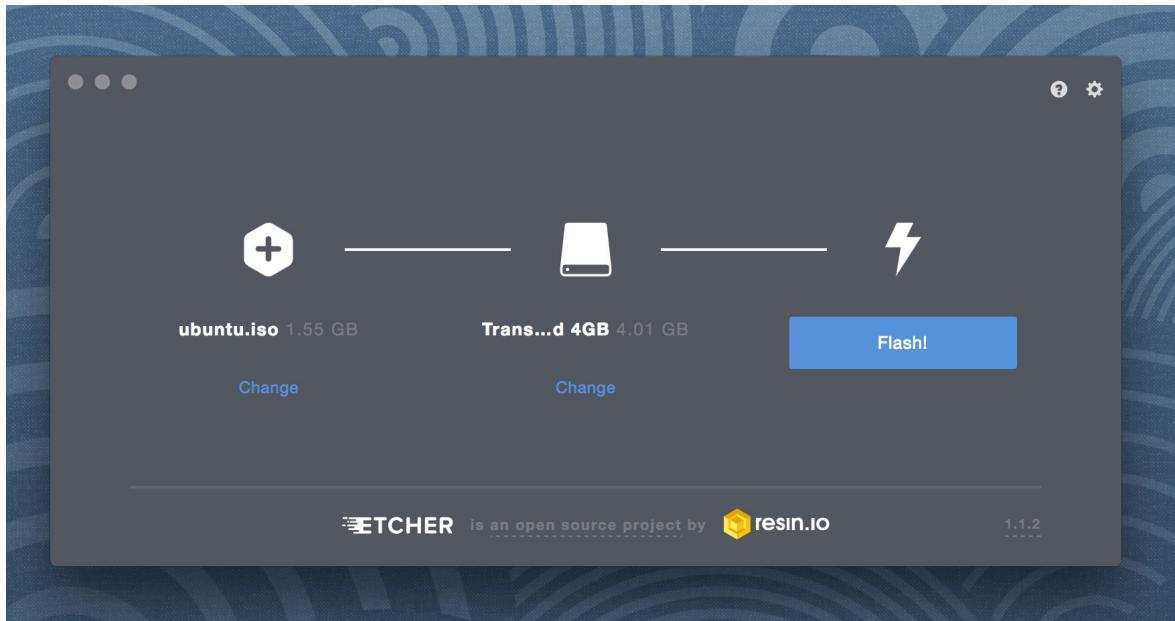


Figure: Etcher complete message

#### 15.5.2.4 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.

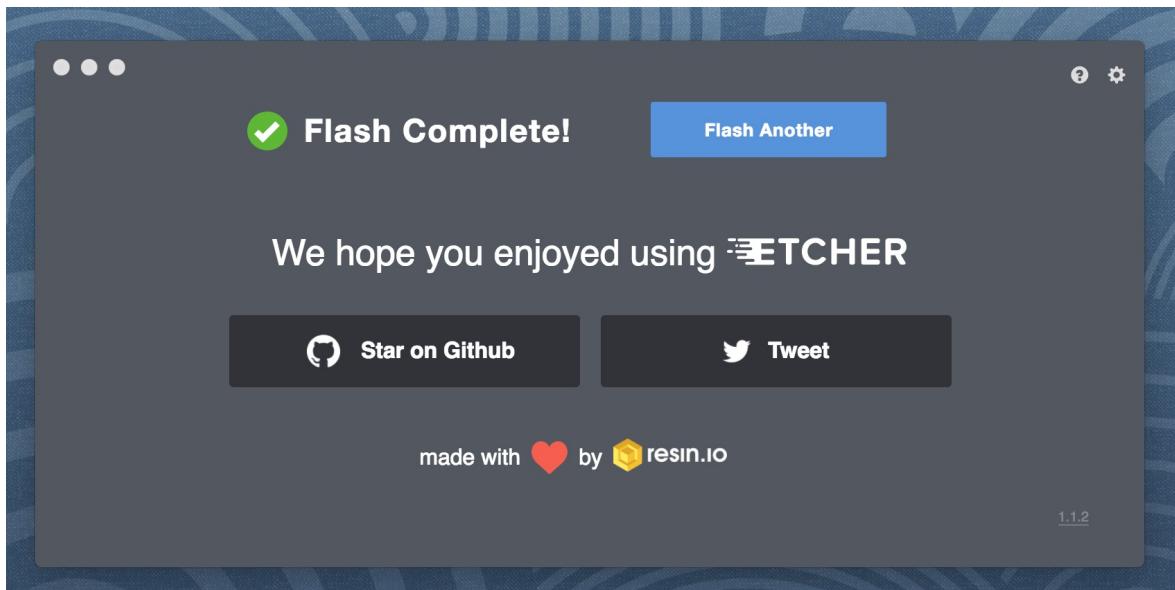


Figure: Etcher

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.

### 15.5.3 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.

### 15.5.4 Exercise

Development.Server.1

If you are in need to boot 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.

## 15.6 GITHUB

---



### 🎓 Learning Objectives

- Be able to use the github cloud sevices 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](#)). Than we will introduce you how you can contribute to this set of documentations with pull requests.

### 15.6.1 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.

## 15.6.2 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 (O 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](#).

## 15.6.3 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](#)

## 15.6.4 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 sometimes 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](#)

#### **15.6.5 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.

Throughout 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>

## **15.6.6 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](#)

## **15.6.7 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](#)

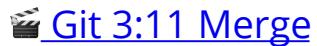
## **15.6.8 Checkout**

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](#)

## **15.6.9 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.



## 15.6.10 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.



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.

## 15.6.11 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>



## 15.6.12 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.

### 15.6.13 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.**

### 15.6.14 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.

### 15.6.15 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 good to still create a branch, work in the branch and then 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
```

## 15.6.16 README.yml and notebook.md

In case you take classes e516 and e616 with us you will have to create a README.yaml 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.yaml 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.

### **15.6.17 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 choose into which account you like to fork the original repository. Next you will create a clone from your forked 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 link 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.

### 15.6.17.1 Stay up to date with the original repo

From 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` that 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.

## 15.6.17.2 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>

## 15.6.18 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.

## 15.6.19 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/>

### 15.6.19.1 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.

## 15.6.19.2 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

as a reference. We like to mention the special enhancements fo github's markdown that integrate well to support project management.

### 15.6.19.2.1 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 of the task text, you need to escape it with a \ )

### 15.6.19.2.2 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/>

#### **15.6.19.2.3 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.

#### **15.6.19.2.4 Emojis**

Although github supports emojis such as :+1: we do not use them typically in our class.

### **15.6.19.3 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.

### **15.6.19.4 cc**

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

### **15.6.19.5 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.

## 15.6.20 GIT PULL REQUEST



### 15.6.20.1 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.

### 15.6.20.2 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.

### 15.6.20.3 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 `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 `cloudmesh/technologies` repository. In our case the user name will be `pulasthi`.

#### 15.6.20.4 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.

#### 15.6.20.5 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 `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)
```

#### 15.6.20.6 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 in Github, not the `cloudmesh/technologies` repo.

### 15.6.20.7 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 a button named `Pull request`

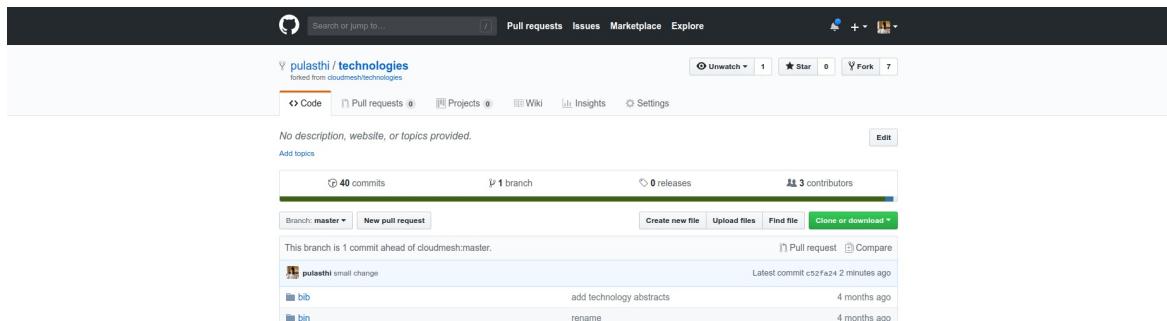


Figure 141: Button Pull request

Once you click on that button you will be taken to a page to create the pull request, which will look similar to [Figure 142](#).

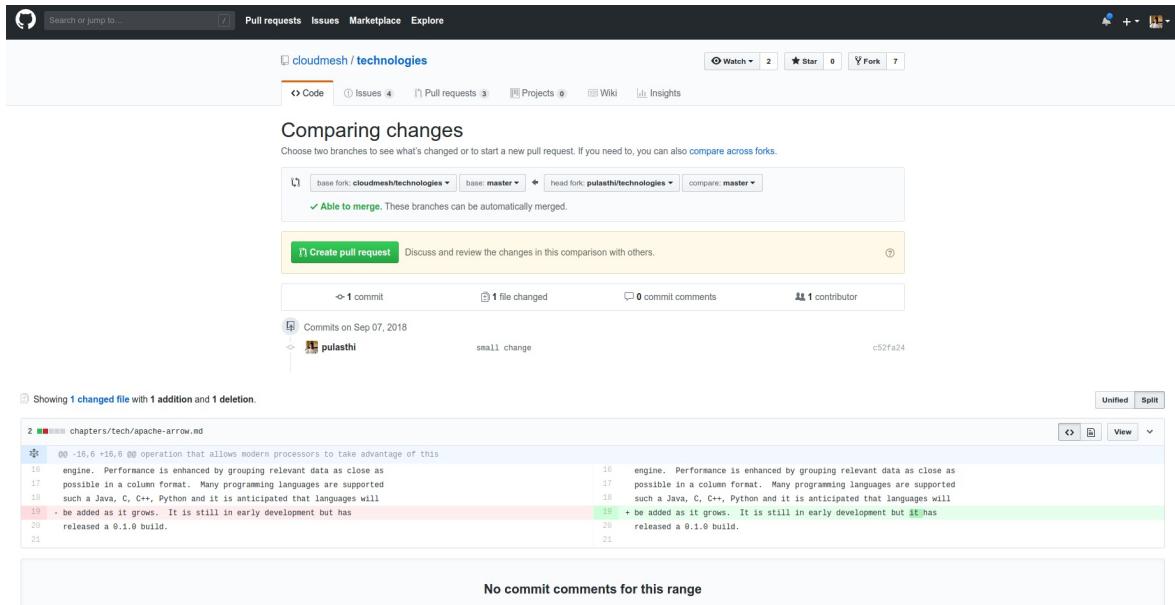


Figure 142: Create a pull request

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**

## 15.7 LINUX



### 🎓 Learning Objectives

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

Now that you have Linux or a Linux like environment (such as `gitbash`)

on your computer it is time to learn a number of useful commands to interact with the system.

In order for this task to enhance your knowledge you are encouraged to find additional material and are required to complete the table of useful Linux commands. You will do this as team and create pull requests improving and completing this documentation. The TAs will provide a mapping between students and commands to be documented. If you find additional commands that ought to be listed here, please add.

### **15.7.1 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.

### **15.7.2 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](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.

.

<b>Command</b>	<b>Description</b>
man command	manual page for the command
apropos text	list all commands that have text in it
ls	Directory listing
ls -lisa	list details
tree	list the directories in graphical form
cd dirname	Change directory to dirname
mkdir dirname	create the directory
rmdir dirname	delete the directory
pwd	print working directory
rm file	remove the file
cp a b	copy file a to b
mv a b	move/rename file a to b
cat a	print content of filea
cat -n filename	print content of filea with line numbers

less a	print paged content of file a
head -5 a	Display first 5 lines of file a
tail -5 a	Display last 5 lines of file a
du -hs .	show in human readable form the space used by the current directory
df -h	show the details of the disk file system
wc filename	counts the word in a file
sort filename	sorts the file
uniq filename	displays only uniq entries in the file
tar -xvf dir	tars up a compressed version of the directory
rsync	faster, flexible replacement for rcp
gzip filename	compresses the file
gunzip filename	compresses the file
bzip2 filename	compresses the file with block-sorting
bunzip2 filename	uncompresses the file with block-sorting
clear	clears the terminal screen
touch filename	change file access and modification times or if file does not exist creates file
	displays a list of users that are currently logged on, for each

who	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
echo -n string	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 command	times the command execution

time command	in seconds
find / [-name] file-name.txt	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 filename	finds the path of a file
grep 'word' filename	finds all lines with the word in it
grep -v 'word' filename	finds all lines without the word in it
chmod ug+rw filename	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	user
su	substitute user identity
uname	print the operating system name
set -o emacs	tells the shell to use Emacs commands.
chmod go-rwx file	changes the permission of the file
chown username file	changes the ownership of the file
chgrp group file	changes the group of a file
fgrep text filename	searches the text in the given file
grep -R text .	recursively searches for xyz in all files
find . -name *.py	find all files with <code>.py</code> 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

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 command	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
traceroute	print the route packets take to 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

ssh	remote login program
scp	remote file copy program
sftp	secure file transfer program
watch command	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 some_file.json   python -m json.tool	quick and easy JSON validator

### 15.7.3 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 &
```

```
command1 &
```

## 15.7.4 Keyboard Shortcuts

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

Keys	Description
Up Arrow	Show the previous command
Ctrl + z	Stops the current command
	Resume with <b>fg</b> in the foreground
	Resume with <b>bg</b> 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

## 15.7.5 bashrc and bash\_profile

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 your `.bash_profile` file:

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

## More Information

<https://cloudmesh.github.io/classes/lesson/linux/refcards.html>

### 15.7.6 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, if 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 sis available at <http://www.cs.jhu.edu/~joanne/unixRC.pdf>.

### 15.7.6.1 Makefiles on Windows

Makefiles can easily be accessed also on windows while installing gitbash. Please reed to the internet or search in this handbook for more information about gitbash.

### 15.7.7 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:

<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:

<b>Permission:</b>	<b>Description:</b>
<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
<code>o</code>	Specifies the permissions of users not in the group

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

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
```

## 15.7.8 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

## 15.8 SECURE SHELL

---



### 🎓 Learning Objectives

- This is one of the most important sections of the book, study it carefully.
  - learn how to use SSH keys
  - Learn how to use ssh-add and ssh-keychain 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 the messages sent 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 then 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.

---

### 15.8.1 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 AAAAB3NzaC1yc2EAAAQABAAQCXJH2iG2FMHqC6T/U7uB8kt
6K1Rh4kU0jgw9sc4UU+Uwe/kshuispauhfsjhfm,anf6787sjgdkjsgl+EwD0
thkoamyi0VvhTVZhj61pTdhyl1t8h1koL19JVnVBPP5kIN3wVyNAJjYBrAUNW
4dXXXtmfkXp98T30W4mxAtTH434MaT+QcPTcxims/hwsUeDAVKZY7UgZhEbIE
xxkejttnRBHTipi0W03W05TOUGRW7EuKf/4ftNVPilC04DpfY44NFG1xPwHeim
Uk+t9h48pBQj16FrUCp0rS02Pj+4/9dNeS1kmNJu5ZYS8HVRhvuoTXuAY/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 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 regenerate 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.
```

## 15.8.2 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.
	Sets the maximum time the agent will keep the given key. After the timeout expires, the key will be

---

-t life	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.
-x	Unlocks the agent. This asks for a password to unlock.
-x	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.

---

### 15.8.3 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
```

### 15.8.3.1 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.

### 15.8.3.2 Using SSH on Linux

All Linux versions come with ssh and can be used right from the terminal.

### 15.8.3.3 Using SSH on Raspberry Pi 3

SSH is available on Raspbian. However, to ssh into the PI you have to activate it via the configuration menu. For a more automated configuration, we will provide more information in the Raspberry PI section.

### 15.8.3.4 SSH on Windows ?

⚠ This section is outdated and should be replaced with information

from SSH in powershell and the new ubuntu running in windows

- <https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands/>

In case you need access to ssh Microsoft has fortunately updated their software to be able to run it directly from the Windows commandline including PowerShell.

However it is as far as we know not activated by default so you need to follow some setup scripts. Also this software is considered beta and its development and issues can be found at

<https://github.com/PowerShell/Win32-OpenSSH>

<https://github.com/PowerShell/Win32-OpenSSH/issues> What you have to do is to install it by going to

Settings > Apps

and click

Manage optional features

under

Apps & features

Next, Click on the `Add feature`. You will be presented with a list in which you scroll down, till you find `openssh client (Beta)`. Click on it and invoke `Install`.

After the install has completed, you can use the `ssh` command. Just type it in the commandshell or PowerShell

PS C:\Users\gregor> ssh

Naturally you can now use it just as on Linux or OSX. and use it to login to other resources

PS C:\Users\gregor> ssh myname@example.com

## 15.8.4 SSH and putty

We no longer recommend the use of putty and instead you should be using SSH over Powershell for this class.

### 15.8.4.1 Access 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
```

## 15.8.5 SSH Port Forwarding

 this section has not been vetted yet

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.

#### 15.8.5.1 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.

#### 15.8.5.2 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.

### 15.8.5.3 Types of Port Forwarding

There are three types of SSH Port forwarding:

#### 15.8.5.4 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.

### 15.8.5.5 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@clodcomputing.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 above 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.

### 15.8.5.6 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@clodcomputing.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.

### 15.8.5.7 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

### 15.8.5.8 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.

### 15.8.5.9 References

- [The Secure Shell: The Definitive Guide, 2 Ed \(O'Reilly and Associates\)](#)

### 15.8.6 SSH to FutureSystems Resources



#### 🎓 Learning Objectives

- Obtain a Future system account so you can use kubernetes or dockerswarm or other services offered by FutureSystems.
- Note that we no longer support OpenStack in 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



#### User account

Create new account  Request new password

Username or e-mail address: \*

You may login with either your assigned username or your e-mail address.

Password: \*

The password field is case sensitive.

[Log in using OpenID](#)

image

Step 2: Click in the “ssh key” button or go directly to <https://portal.futuresystems.org/my/ssh-keys>

Staff Welcome, jdiaz! Logout Search FutureGrid... [f](#) [t](#) [r](#)

Summer School About News Support Community Projects

## My Portal Account

View Portal Account Bookmarks Edit Messages Notifications OpenID identities Publications Subscriptions Track Broken links File browser SSH keys

**My profile info**

Profile Picture Contact

image

Step 3: Click in the “add a public key” link.

Staff Welcome, jdiaz! Logout Search FutureGrid... [f](#) [t](#) [r](#)

Summer School About News Support Community Projects

## My account

View Portal Account Bookmarks Edit Messages Notifications OpenID identities Publications Subscriptions Track Broken links File browser SSH keys

Need help with public keys? View the excellent GitHub.com SSH public key help at <http://github.com/guides/providing-your-ssh-key>.

- Add a public key

Title	Fingerprint	Operations
javinew	71:6e:5a:a4:7d:45:4e:5b:55:e2:3e:b0:43:f7:c5:ed	Edit Delete
jdiaz-india	fe:8e:8c:98:f3:49:02:56:f1:a0:7e:21:46:b9:4a:7b	Edit Delete
jdiaz@javi-OptiPlex-960	d9:96:06:b4:cf:0f:5d:79:63:fb:38:60:61:15:30:7c	Edit Delete
jdiaz@localhost	42:af:52:fa:01:dc:4c:14:82:ea:0d:8b:02:eb:be:dc	Edit Delete
minicluster	c5:3f:01:fb:b3:e1:6e:e8:f5:a6:7f:04:3d:1e:af:45	Edit Delete
rsa-key-20101004	c6:f5:05:0b:bf:09:4a:31:ef:f4:d3:65:4c:ca:68:83	Edit Delete
sc11-key	46:8b:8f:44:75:a3:16:21:61:63:96:01:82:e3:36:73	Edit Delete
sierra	23:b2:97:39:26:4c:da:c7:38:9a:75:3b:52:c6:c3:d8	Edit Delete

image

Step 4: Paste your ssh key into the box marked 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.

FutureGrid  
Portal

Staff Welcome, jdiaz! Logout Search FutureGrid...

[Summer School](#) [About](#) [News](#) [Support](#) [Community](#) [Projects](#)

## Add a SSH key

Need help with public keys? View the excellent GitHub.com SSH public key help at <http://github.com/guides/providing-your-ssh-key>.

**Title:**

If this field is left blank, the key's title will be automatically generated.

**Key: \***

```
ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCXJH2iG2FMHqC6T/U7uB8kt6KIRh4kUOjgw9sc4Uu+Uwe/EwD0wk6CBQMB+HKb9upvCRW/851UyRUagt
IQexCRM2rMci0VvhTVZhj61pTdhyl1t8hkl0L19JvnVBPP5kIn3wVyNAjYBrAUNW4dXXtmfkXp98T3OW4mxAtTH434MaT+QcPTcxims/hwsUeDAV
KZY7UgZhEbiExxkejtRBHTipi0W03W05TOUGRW7EuKf/4ftNVPiCO4DpfY44NFG1xPwHeimUk+t9h48pBQj16FrUCp0rS02Pj+4/9dNeS1kmNJU5ZY
S8HVRhvu0TxuAY/UVcynEPUegkp+qYnR Javi@Javi-PC
```

image

Step 5: Click the submit button. **IMPORTANT:** Leave the Title field blank. Make sure that when you paste your key, it does not contain newlines or carriage returns that may have been introduced by incorrect pasting and copying. If so, please remove them.

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.

### 15.8.6.1 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.

## 15.8.7 Exercises



E.SSH.1:

Create an SSH key pair

E.SSH.2:

Upload the public key to git repository you use. Create a fork in git and use your ssh key to clone and commit to it section/ssh.tex

E.SSH.3:

The images in the futuresystems ssh section are a bit outdated. Please update them. Make sure to blend out your username and fingerprints in the images. or invent some ...

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.



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 intertwined 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 challenge 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 from <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 venn diagram for visualizing the important aspects that DevOps tries to address is depicted in [Figure 143](#)

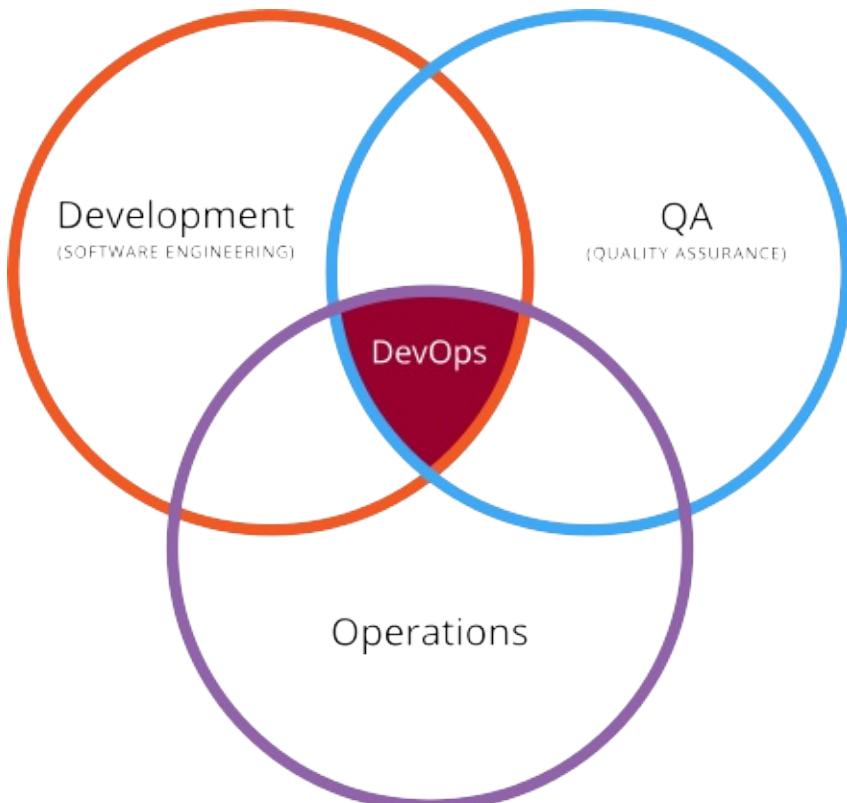


Figure 143: Wikipedia Ven Diagram about DevOps

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 144](#)



Figure 144: Wikipedia Ven Diagram about DevOps

The cycle includes in each portion

Dev:

- dev planning
- creating
- verifying
- packaging

Ops:

- releasing
- configuring
- monitoring
- ops planning (influencing dev planning)

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.

## **16.1 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/>

## **16.2 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.

### 16.2.1 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.

## 16.2.2 Resources

- <https://docs.travis-ci.com/>

## 16.3 ANSIBLE

---



### 16.3.1 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.

#### 16.3.1.1 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.

### **16.3.1.2 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 your 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.

### 16.3.1.3 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.

### 16.3.2 Ansible Roles

Next we install the R package onto our cloud VMs. R is a useful statistic programing 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 above.

### 16.3.3 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.

### 16.3.4 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.

#### 16.3.4.1 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 145](#) shows how it looks like.

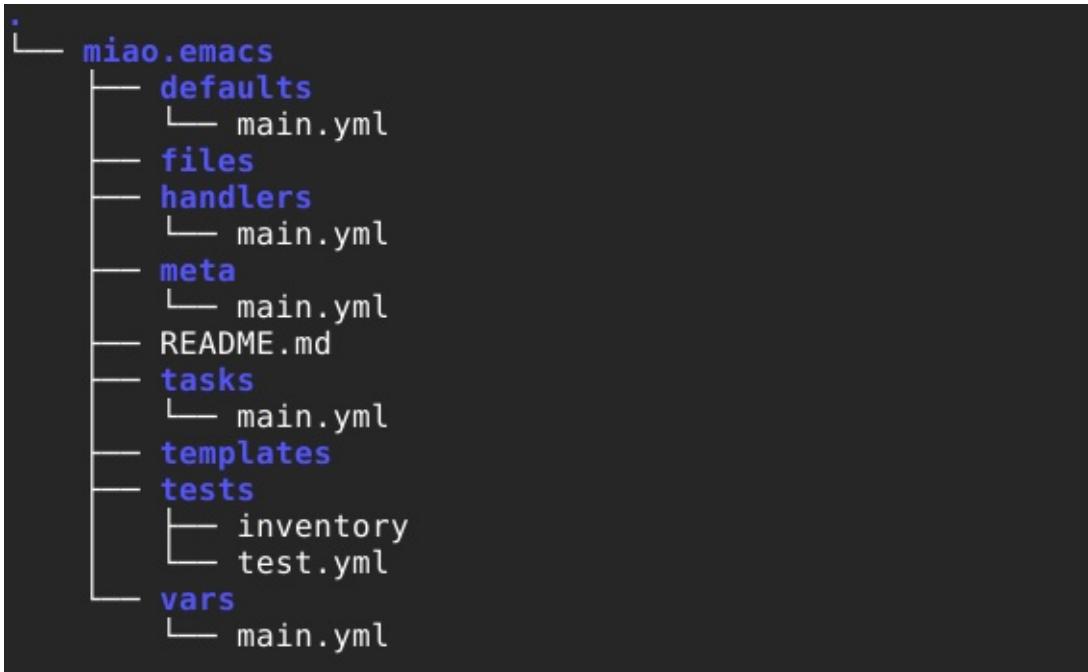


Figure 145: image

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.

### 16.3.5 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

### **16.3.5.1 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.

### **16.3.5.2 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/\\*](http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/*)

We also assume that we install MongoDB on Ubuntu 15.10.

#### 16.3.5.2.1 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

```

### 16.3.5.2.2 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.

### 16.3.5.2.3 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](http://docs.ansible.com/list_of_all_modules.html)

#### 16.3.5.2.4 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
```

#### 16.3.5.2.5 Module apt\_repository: add repositories

Next add the MongoDB repository to apt:

```
- name: Add MongoDB repository
 apt_repository: repo='deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10.04 main'
 ↻
```

#### 16.3.5.2.6 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
```

#### 16.3.5.2.7 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
```

#### 16.3.5.2.8 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'
- name: install mongodb
 apt: pkg=mongodb-org state=latest update_cache=yes
 notify:
 - start mongodb
 handlers:
 - name: start mongodb
 service: name=mongod state=started

```

### 16.3.5.2.9 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

```

### 16.3.5.2.10 Sanity Check: Test MongoDB

Let's 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
>

```

### 16.3.5.2.11 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.

#### **16.3.5.2.12 Reference**

The main tutorial from Ansible is here:  
[http://docs.ansible.com/playbooks\\_intro.html](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](http://docs.ansible.com/modules_by_category.html)

#### **16.3.6 Exercise**

We have shown a couple of examples of using Ansible tools. Before you apply it in your 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



## 17.1 INTRODUCTION TO PYTHON



### 🎓 Learning Objectives

- Learn quickly Python under the assumption you know a programming language
- Work with modules
- Understand docopts and cmd
- Contact some python examples to refresh your python knowledge
- Learn about the `map` function in Python
- Learn how to start subprocesses and redirect their output
- Learn more advanced constructs such as multiprocessing and Queues
- Understand why we do not use `anaconda`
- Get familiar with `pyenv`

---

Portions of this lesson have been adapted from the [official Python Tutorial](#) copyright [Python Software Foundation](#).

Python is an easy to learn programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's simple syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter can be extended with new functions and data types implemented in C or C++

(or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python is an interpreted, dynamic, high-level programming language suitable for a wide range of applications.

The philosophy of python is summarized in [The Zen of Python](#) as follows:

- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

The main features of Python are:

- Use of indentation whitespace to indicate blocks
- Object orient paradigm
- Dynamic typing
- Interpreted runtime
- Garbage collected memory management
- a large standard library
- a large repository of third-party libraries

Python is used by many companies (such as Google, Yahoo!, CERN, NASA) and is applied for web development, scientific computing, embedded applications, artificial intelligence, software development, and information security, to name a few.

The material collected here introduces the reader to the basic concepts and features of the Python language and system. After you have worked through the material you will be able to:

- use Python
- use the interactive Python interface
- understand the basic syntax of Python
- write and run Python programs stored in a file
- have an overview of the standard library

- install Python libraries using pyenv or if it is not available virtualenv

This section does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules.

In order to conduct this lesson you need

- A computer with Python 2.7.15 or 3.7.0
- Familiarity with command line usage
- A text editor such as [PyCharm](#), emacs, vi or others. You should identify which works best for you and set it up.

### 17.1.1 References

Some important additional information can be found on the following Web pages.

- [Python](#)
- [Pip](#)
- [Virtualenv](#)
- [NumPy](#)
- [SciPy](#)
- [Matplotlib](#)
- [Pandas](#)
- [pyenv](#)
- [PyCharm](#)

Python module of the week is a Web site that provides a number of short examples on how to use some elementary python modules. Not all modules are equally useful and you should decide if there are better alternatives. However for beginners this site provides a number of good examples

- Python 2: <https://pymotw.com/2/>
- Python 3: <https://pymotw.com/3/>

## 17.2 PYTHON INSTALLATION



Python is easy to install and very good instructions for most platforms can be found on the python.org Web page. We will be using Python 2.7.15 and/or Python 3.7 in our activities.

To manage python modules, it is useful to have [pip](#) package installation tool on your system.

We assume that you have a computer with python installed. The version of python however may not be the newest version. Please check with

```
$ python --version
```

which version of python you run. If it is not the newest version, we recommend that you install pyenv to install a newer version so you do not effect the default version of python from your system.

While in other classes yo may have been taught to use anaconda, this is not a tool that ought to be used in a cloud class. The reason for this is that it installs many packages that you are likely not to use. In fact installing anaconda on your VM will waste space and time and you should look into other installs.

However, real cloud engineers with the most flexibility in python versions want to install python via pyenv.

Note: whenever possible please use for the newest version of Python 2 or 3. In order not to effect your OS we will use pyenv.

### 17.2.1 Managing custom Python installs

Often you have your own computer and you do not like to change its environment to keep it in pristine condition. Python comes with many

libraries that could for example conflict with libraries that you have installed. To avoid this it is best to work in an isolated python we can use tools such as virtualenv, pyenv or pyenv for 3.7.1. Which you use depends on you, but we highly recommend pyenv if you can.

### **17.2.1.1 Managing Multiple Python Versions with Pyenv**

Python has several versions that are used by the community. This includes Python 2 and Python 3, but all different management of the python libraries. As each OS may have their own version of python installed. It is not recommended that you modify that version. Instead you may want to create a localized python installation that you as a user can modify. To do that we recommend pyenv. Pyenv allows users to switch between multiple versions of Python (<https://github.com/yyuu/pyenv>). To summarize:

- users to change the global Python version on a per-user basis;
- users to enable support for per-project Python versions;
- easy version changes without complex environment variable management;
- to search installed commands across different python versions;
- integrate with tox (<https://tox.readthedocs.io/>).

#### **17.2.1.1.1 Installation without pyenv**

If you need to have more than one python version installed and do not want or can use pyenv, we recommend you download and install python 2.7.15 and 3.7.1 from python.org (<https://www.python.org/downloads/>)

#### **17.2.1.1.2 Disabling wrong python installs on macOS**

While working with students we have seen at times that they take other classes either at universities or online that teach them how to program in python. Unfortunately, although they seem to do that

they often ignore to teach you how to properly install python. I just recently had a students that had installed python 7 times on his macOS machine, while another student had 3 different installations, all of which conflicted with each other as they were not set up properly.

We recommend that you inspect if you have files such as `~/.bashrc` or `~/.bashrc_profile` in your home directory and identify if it activates various versions of python on your computer. If so you could try to deactivate them while out-commenting the various versions with the `#` character at the beginning of the line, start a new terminal and see if the terminal shell still works. Then you can follow our instructions here while using an install on pyenv.

#### **17.2.1.1.3 Install pyenv on macOS from git**

This is our recommended way to install pyenv on macOS:

```
$ git clone https://github.com/pyenv/pyenv.git ~/.pyenv
$ git clone https://github.com/pyenv/pyenv-virtualenv.git ~/.pyenv/plugins/pyenv-virtualenv
$ git clone https://github.com/yyuu/pyenv-virtualenvwrapper.git ~/.pyenv/plugins/pyenv-virt
$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bash_profile
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bash_profile
```

#### **17.2.1.1.4 Installation of Homebrew**

Before installing anything on your computer make sure you have enough space. Use in the terminal the command:

```
$ df -h
```

which gives you an overview of your file system. If you do not have enough space, please make sure you free up unused files from your drive.

In many occasions it is beneficial to use readline as it provides nice editing features for the terminal and xz for completion. First, make sure you have xcode installed:

```
$ xcode-select --install
```

On Mojave you will get an error that zlib is not installed. This is due to that the header files are not properly installed. To do this you can say

```
$ sudo installer -pkg /Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macos_10.14.x_arm64.pkg -target /
```

Next install homebrew, pyenv, pyenv-virtualenv and pyenv-virtualwrapper. Additionally install readline and some compression tools:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

```
brew update
```

```
brew install readline xz
```

#### 17.2.1.5 Install pyenv on macOS with Homebrew

This is the recommended way of installing pyenv on macOS High Sierra. This method should also be considered if you get the following error: “ERROR: The Python ssl extension was not compiled. Missing the OpenSSL lib?”

We describe here a mechanism of installing pyenv with homebrew. Other mechanisms can be found on the pyenv documentation page (<https://github.com/yyuu/pyenv-installer>). You must have homebrew installed as discussed in the previous section.

To install pyenv with homebrew execute in the terminal:

```
brew install pyenv pyenv-virtualenv pyenv-virtualenvwrapper
```

#### 17.2.1.6 Install pyenv on Ubuntu

The following steps will install pyenv in a new ubuntu 18.04 distribution.

Start up a terminal and execute in the terminal the following commands. We recommend that you do it one command at a time so you can observe if the command succeeds:

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

```
$ sudo apt-get install git python-pip make build-essential libssl-dev
```

```
$ sudo apt-get install zlib1g-dev libbz2-dev libreadline-dev libssqlite3-dev
$ sudo pip install virtualenvwrapper

$ git clone https://github.com/yyuu/pyenv.git ~/.pyenv
$ git clone https://github.com/pyenv/pyenv-virtualenv.git ~/.pyenv/plugins/pyenv-virtualenv
$ git clone https://github.com/yyuu/pyenv-virtualenvwrapper.git ~/.pyenv/plugins/pyenv-virtualenvwrapper

$ echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
$ echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
```

You can also install pyenv using curl command in following way:

```
curl -L https://raw.githubusercontent.com/yyuu/pyenv-installer/master/bin/pyenv-installer
```

Then install its dependencies:

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install -y make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-
```

Now that you have installed pyenv it is not yet activated in your current terminal. The easiest thing to do is to start a new terminal and type in:

```
which pyenv
```

If you see a response pyenv is installed and you can proceed with the next steps.

Please remember whenever you modify `.bashrc` or `.bash_profile` you need to start a new terminal.

#### 17.2.1.1.7 Install Different Python Versions

Pyenv provides a large list of different python versions. To see the entire list please use the command:

```
$ pyenv install -l
```

However, for us we only need to worry about python 2.7.15 and python 3.7.1. You can now install different versions of python into your local environment with the following commands:

```
$ pyenv update
$ pyenv install 2.7.15
$ pyenv install 3.7.1
```

You can set the global python default version with:

```
$ pyenv global 3.7.1
```

Type the following to determine which version you activated:

```
$ pyenv version
```

Type the following to determine which versions you have available:

```
$ pyenv versions
```

Associate a specific environment name with a certain python version, use the following commands:

```
$ pyenv virtualenv 2.7.15 ENV2
$ pyenv virtualenv 3.7.1 ENV3
```

In the example, ENV2 would represent python 2.7.15 while ENV3 would represent python 3.7.1. Often it is easier to type the alias rather than the explicit version.

#### 17.2.1.1.8 Set up the Shell

To make all work smoothly from your terminal, you can include the following in your `.bashrc` file in case you run Linux. For macOS it is best to place it in your `.bash_profile` file:

```
export PYENV_VIRTUALENV_DISABLE_PROMPT=1
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"

__pyenv_version_ps1() {
 local ret=$?;
 output=$(pyenv version-name)
 if [[! -z $output]]; then
 echo -n "($output)"
 fi
 return $ret;
}

PS1="\$(__pyenv_version_ps1) ${PS1}"
```

We recommend that you do this towards the end of your file.

#### 17.2.1.1.9 Switching Environments

After setting up the different environments, switching between them is now easy. Simply use the following commands:

```
(2.7.15) $ pyenv activate ENV2
(ENV2) $ pyenv activate ENV3
(ENV3) $ pyenv activate ENV2
(ENV2) $ pyenv deactivate ENV2
(2.7.15) $
```

To make it even easier, you can add the following lines to your `.bash_profile` file:

```
alias ENV2="pyenv activate ENV2"
alias ENV3="pyenv activate ENV3"
```

If you start a new terminal, you can switch between the different versions of python simply by typing:

```
$ ENV2
$ ENV3
```

#### 17.2.2 Updating Python Version List

Pyenv maintains locally a list of available python versions. To see the list use the command

```
pyenv update
pyenv install -l
```

You will see the updated list.

#### 17.2.3 Updating to a new version of Python with pyenv

Naturally python itself evolves and new versions will become available via pyenv. To facilitate such a new version you need to first install it into pyenv. Let us assume you had an old version of python installed onto the ENV3 environment. Then you need to execute the following steps:

```
pyenv deactivate
pyenv uninstall ENV3
pyenv install 3.7.1
pyenv virtualenv 3.7.1 ENV3
ENV3
pip install pip -U
```

With the pi install command, we make sure we have the newest version of pip. In case you get an error, you may have to update xcode as follows and try again:

```
xcode-select --install
```

After you installed it you can activate it by typing `ENV3`. Naturally this requires that you added it to your bash environment as discussed in Section [1.1.1.8](#).

## 17.2.4 Pyenv in a docker container

We provide a simple docker container on docker hub that is based on ubuntu 18.04 that has pyenv, python 2.7.15 and python 3.7.1 installed. Using this image is as simple as downloading it and running it.

To run the container and log into the command prompt please use

```
$ docker run --rm -it cloudmesh/pyenv:1.0 /bin/bash
```

To switch between the python versions use the command

```
container> ENV2
container> ENV3
```

where `container` indicates that the command is executed ###  
Creating the container locally

This section is only needed if you like to recreate the image or modify the Dockerfile.

The information about how we create the image is provided at in a repository. You can download the code in the directory and can create the image from the Docker file while using the Makefile as

follows:

```
$ mkdir cloudmesh-community
$ cd cloudmesh-community
$ git clone https://github.com/cloudmesh-community/images.git
$ cd images/pyenv
$ make image
```

This will create an image locally. with

```
$ make login
```

you can login to the shell. Typically you will only need the docker command as described in the previous section.

## 17.2.5 Installation without pyenv

If you need to have more than one python version installed and do not want or can use pyenv, we recommend you download and install python 2.7.15 and 3.7.1 from python.org (<https://www.python.org/downloads/>)

### 17.2.5.1 Make sure pip is up to date

As you will want to install other packages, make sure pip is up to date:

```
pip install pip -U
```

```
pyenv virtualenv anaconda3-4.3.1 ANA3
pyenv activate ANA3
```

## 17.2.6 Anaconda and Miniconda

We do not recommend that you use anaconda or miniconda as it may interfere with your default python interpreters and setup.

Please note that beginners to python should always use anaconda or miniconda only after they have installed pyenv and use it. For this class neither anaconda nor miniconda is required. In fact we do not recommend it. We keep this section as we know that other classes at IU may use anaconda. We are not aware if these classes teach you

the right way to install it, with pyenv.

### 17.2.6.1 Miniconda

⚠ This section about miniconda is experimental and has not been tested. We are looking for contributors that help completing it. If you use anaconda or miniconda we recommend to manage it via pyenv.

To install mini conda you can use the following commands:

```
$ mkdir ana
$ cd ana
$ pyenv install miniconda3-latest
$ pyenv local miniconda3-latest
$ pyenv activate miniconda3-latest
$ conda create -n ana anaconda
```

To activate use:

```
$ source activate ana
```

To deactivate use:

```
$ source deactivate
```

To install cloudmesh cmd5 please use:

```
$ pip install cloudmesh.cmd5
$ pip install cloudmesh.sys
```

### 17.2.6.2 Anaconda

⚠ This section about anaconda is experimental and has not been tested. We are looking for contributors that help completing it.

You can add anaconda to your pyenv with the following commands:

```
pyenv install anaconda3-4.3.1
```

To switch more easily we recommend that you use the following in your `.bash_profile` file:

```
alias ANA="pyenv activate anaconda3-4.3.1"
```

Once you have done this you can easily switch to anaconda with the command:

```
$ ANA
```

Terminology in anaconda could lead to confusion. Thus we like to point out that the version number of anaconda is unrelated to the python version. Furthermore, anaconda uses the term root not for the root user, but for the originating directory in which the anaconda program is installed.

In case you like to build your own conda packages at a later time we recommend that you install the conda-build package:

```
$ conda install conda-build
```

When executing:

```
pyenv versions
```

you will see after the install completed the anaconda versions installed:

```
pyenv versions
system
2.7.15
2.7.15/envs/ENV2
3.7.1
3.7.1/envs/ENV3
ENV2
ENV3
* anaconda3-4.3.1 (set by PYENV_VERSION environment variable)
```

Let us now create virtualenv for anaconda:

```
$ pyenv virtualenv anaconda3-4.3.1 ANA
```

To activate it you can now use:

```
$ pyenv ANA
```

However, anaconda may modify your `.bashrc` or `.bash_profile` files and may result in incompatibilities with other python versions. For this reason we recommend not to use it. If you find ways to get it to work

reliably with other versions, please let us know and we update this tutorial.

To install cloudmesh cmd5 please use:

```
$ pip install cloudmesh.cmd5
$ pip install cloudmesh.sys
```

### 17.2.6.3 virtualenv

Documentation about it can be found at:

```
* https://virtualenv.pypa.io
```

The installation is simple once you have pip installed. If it is not installed you can say:

```
$ easy_install pip
```

After that you can install the virtual env with:

```
$ pip install virtualenv
```

To setup an isolated environment for example in the directory ~/ENV please use:

```
$ virtualenv ~/ENV
```

To activate it you can use the command:

```
$ source ~/ENV/bin/activate
```

you can put this command in your `.bashrc` or `.bash_profile` files so you do not forget to activate it. Instructions for this can be found in our lesson on Linux `bashrc`.

#### 17.2.6.3.1 Exercises

E.Python.Install.0:

Write installation instructions for an operating system of your choice and add to this documentation.

E.Python.Install.1:

Replicate the steps to install pyenv, so you can type in ENV2 and ENV3 in your terminals to switch between python 2 and 3.

E.Python.Install.3:

Why do you not want to use generally anaconda for cloud computing? When is it ok to use anaconda?

## 17.3 INTERACTIVE PYTHON



Python can be used interactively. You can enter the interactive mode by entering the interactive loop by executing the command:

```
$ python
```

You will see something like the following:

```
python
Python 3.7.1 (default, Nov 24 2018, 14:27:15)
[Clang 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The `>>>` is the prompt used by the interpreter. This is similar to bash where commonly `$` is used.

Sometimes it is convenient to show the prompt when illustrating an example. This is to provide some context for what we are doing. If you are following along you will not need to type in the prompt.

This interactive python process does the following:

- read your input commands
- evaluate your command
- print the result of evaluation
- loop back to the beginning.

This is why you may see the interactive loop referred to as a **REPL**:

**Read-Evaluate-Print-Loop.**

### 17.3.1 REPL (Read Eval Print Loop)

There are many different types beyond what we have seen so far, such as **dictionaries**, **lists**, **sets**. One handy way of using the interactive python is to get the type of a value using type():

```
>>> type(42)
<type 'int'>
>>> type(hello)
<type 'str'>
>>> type(3.14)
<type 'float'>
```

You can also ask for help about something using help():

```
>>> help(int)
>>> help(list)
>>> help(str)
```

Using help() opens up a help message within a pager. To navigate you can use the spacebar to go down a page w to go up a page, the arrow keys to go up/down line-by-line, or q to exit.

### 17.3.2 Interpreter

Although the interactive mode provides a convenient tool to test things out you will see quickly that for our class we want to use the python interpreter from the commandline. Let us assume the program is called `prg.py`. Once you have written it in that file you simply can call it with

```
$ python prg.py
```

It is important to name the program with meaningful names.

### 17.3.3 Python 3 Features in Python 2

In this course we want to be able to seamlessly switch between python 2 and python 3. Thus it is convenient from the start to use python 3 syntax when it is supported also in python 2. One of the

most used functions is the print statement that has in python 3 parentheses. To enable it in python 2 you just need to import this function:

```
from __future__ import print_function, division
```

The first of these imports allows us to use the print function to output text to the screen, instead of the print statement, which Python 2 uses. This is simply a [design decision](#) that better reflects Python's underlying philosophy.

Other functions such as the division also behave differently. Thus we use

```
from __future__ import division
```

This import makes sure that the [division operator](#) behaves in a way a newcomer to the language might find more intuitive. In Python 2, division / is floor division when the arguments are integers, meaning that the following

```
(5 / 2 == 2) is True
```

In Python 3, division / is a floating point division, thus

```
(5 / 2 == 2.5) is True
```

## 17.4 EDITORS



This section is meant to give an overview of the python editing tools needed for doing for this course. There are many other alternatives, however, we do recommend to use PyCharm.

### 17.4.1 Pycharm

PyCharm is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with git.

 [Python 8:56 Pycharm](#)

## 17.4.2 Python in 45 minutes

An additional community video about the Python programming language that we found on the internet. Naturally there are many alternatives to this video, but the video is probably a good start. It also uses PyCharm which we recommend.

 [Python 43:16 PyCharm](#)

How much you want to understand of python is actually a bit up to you. While its good to know classes and inheritance, you may be able for this class to get away without using it. However, we do recommend that you learn it.

PyCharm Installation: Method 1: PyCharm Installation on ubuntu using umake

```
sudo add-apt-repository ppa:ubuntu-desktop/ubuntu-make
sudo apt-get update
sudo apt-get install ubuntu-make
```

Once umake command is run, use below command to install Pycharm community edition:

```
umake ide pycharm
```

If you want to remove PyCharm installed using umake command, use this:

```
umake -r ide pycharm
```

Method 2: PyCharm installation on ubuntu using PPA

```
sudo add-apt-repository ppa:mystic-mirage/pycharm
sudo apt-get update
sudo apt-get install pycharm-community
```

PyCharm also has a Professional (paid) version which can be installed using following command:

```
sudo apt-get install pycharm
```

Once installed, go to your VM dashboard and search for PyCharm.

## 17.5 LANGUAGE



### 17.5.1 Statements and Strings

TODO: some of the python examples assume REPL, but its better to use a print statement instead as more general, please fix} Let us explore the syntax of Python. Type into the interactive loop and press Enter:

```
print("Hello world from Python!")
```

This will print on the terminal

```
Hello world from Python
```

What happened? The print function was given a **string** to process. A string is a sequence of characters. A **character** can be a alphabetic (A through Z, lower and upper case), numeric (any of the digits), white space (spaces, tabs, newlines, etc), syntactic directives (comma, colon, quotation, exclamation, etc), and so forth. A string is just a sequence of the character and typically indicated by surrounding the characters in double quotes.

Standard output is discussed in the [Section Linux](#).

So, what happened when you pressed Enter? The interactive Python program read the line `print ("Hello world from Python!")`, split it into the print statement and the `"Hello world from Python!"` string, and then executed the line, showing you the output.

### 17.5.2 Variables

You can store data into a **variable** to access it later. For instance, instead of:

```
print('Hello world from Python!')
```

which is a lot to type if you need to do it multiple times, you can store the string in a variable for convenient access:

```
hello = 'Hello world from Python!'
print(hello)
```

This will print again

```
Hello world from Python!
```

## 17.5.3 Data Types

### 17.5.3.1 Booleans

A **boolean** is a value that indicates truthness of something. You can think of it as a toggle: either “on” or “off”, “one” or “zero”, “true” or “false”. In fact, the only possible values of the **boolean** (or `bool`) type in Python are:

- True
- False

You can combine booleans with **boolean operators**:

- and
- or

```
print(True and True)
True

print(True and False)
False

print(False and False)
False

print(True or True)
True

print(True or False)
True

print(False or False)
```

```
False
```

### 17.5.3.2 Numbers

The interactive interpreter can also be used as a calculator. For instance, say we wanted to compute a multiple of 21:

```
print(21 * 2)
42
```

We saw here the print statement again. We passed in the result of the operation  $21 * 2$ . An **integer** (or **int**) in Python is a numeric value without a fractional component (those are called **floating point** numbers, or **float** for short).

The mathematical operators compute the related mathematical operation to the provided numbers. Some operators are:

Operator	Function
*	multiplication
/	division
+	addition
-	subtraction
**	exponent

Exponentiation  $x^y$  is written as  $x^{**}y$  is  $x$  to the  $y$ th power.

You can combine **floats** and **ints**:

```
print(3.14 * 42 / 11 + 4 - 2)
13.9890909091

print(2**3)
8
```

Note that **operator precedence** is important. Using parenthesis to indicate affect the order of operations gives a difference results, as expected:

```
print(3.14 * (42 / 11) + 4 - 2)
11.42

print(1 + 2 * 3 - 4 / 5.0)
6.2

print((1 + 2) * (3 - 4) / 5.0)
-0.6
```

## 17.5.4 Module Management

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. A module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

### 17.5.4.1 Import Statement

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. The from...import Statement Python's from statement lets you import specific attributes from a module into the current namespace. It is preferred to use for each import its own line such as:

```
import numpy
import matplotlib
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module.

### 17.5.4.2 The from ... import Statement

Python's from statement lets you import specific attributes from a module into the current namespace. The from ... import has the following syntax:

```
from datetime import datetime
```

## 17.5.5 Date Time in Python

The `datetime` module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation. For related functionality, see also the `time` and `calendar` modules.

**The import Statement** You can use any Python source file as a module by executing an `import` statement in some other Python source file.

```
from datetime import datetime
```

This module offers a generic date/time string parser which is able to parse most known formats to represent a date and/or time.

```
from dateutil.parser import parse
```

`pandas` is an open source Python library for data analysis that needs to be imported.

```
import pandas as pd
```

Create a string variable with the class start time

```
fall_start = '08-21-2018'
```

Convert the string to datetime format

```
datetime.strptime(fall_start, '%m-%d-%Y') \#
datetime.datetime(2017, 8, 21, 0, 0)
```

Creating a list of strings as dates

```
class_dates = ['8/25/2017', '9/1/2017', '9/8/2017', '9/15/2017', '9/22/2017', '9/29/2017']
```

Convert `Class_dates` strings into datetime format and save the list into variable `a`

```
a = [datetime.strptime(x, '%m/%d/%Y') for x in class_dates]
```

Use `parse()` to attempt to auto-convert common string formats. Parser must be a string or character stream, not list.

```
parse(fall_start)
datetime.datetime(2017, 8, 21, 0, 0)
```

Use `parse()` on every element of the `Class_dates` string.

```
[parse(x) for x in class_dates]
[datetime.datetime(2017, 8, 25, 0, 0),
datetime.datetime(2017, 9, 1, 0, 0),
datetime.datetime(2017, 9, 8, 0, 0),
datetime.datetime(2017, 9, 15, 0, 0),
datetime.datetime(2017, 9, 22, 0, 0),
datetime.datetime(2017, 9, 29, 0, 0)]
```

Use `parse`, but designate that the day is first.

```
parse (fall_start, dayfirst=True)
datetime.datetime(2017, 8, 21, 0, 0)
```

Create a `Dataframe`. A `DataFrame` is a tabular data structure comprised of rows and columns, akin to a spreadsheet, database table. `DataFrame` as a group of `Series` objects that share an index (the column names).

```
import pandas as pd
data = {
 'dates': [
 '8/25/2017 18:47:05.069722',
 '9/1/2017 18:47:05.119994',
 '9/8/2017 18:47:05.178768',
 '9/15/2017 18:47:05.230071',
 '9/22/2017 18:47:05.230071',
 '9/29/2017 18:47:05.280592'],
 'complete': [0, 1, 1, 1, 0, 1]}
df = pd.DataFrame(
 data,
 columns = ['dates','complete'])
print(df)
dates complete
0 8/25/2017 18:47:05.069722 1
1 9/1/2017 18:47:05.119994 0
2 9/8/2017 18:47:05.178768 1
3 9/15/2017 18:47:05.230071 1
4 9/22/2017 18:47:05.230071 0
5 9/29/2017 18:47:05.280592 1
```

Convert `df['date']` from string to `datetime`

```
import pandas as pd
```

```
pd.to_datetime(df['dates'])
0 2017-08-25 18:47:05.069722
1 2017-09-01 18:47:05.119994
2 2017-09-08 18:47:05.178768
3 2017-09-15 18:47:05.230071
4 2017-09-22 18:47:05.230071
5 2017-09-29 18:47:05.280592
Name: dates, dtype: datetime64[ns]
```

## 17.5.6 Control Statements

### 17.5.6.1 Comparison

Computer programs do not only execute instructions. Occasionally, a choice needs to be made. Such as a choice is based on a condition. Python has several conditional operators:

Operator	Function
>	greater than
<	smaller than
==	equals
!=	is not

Conditions are always combined with variables. A program can make a choice using the if keyword. For example:

```
x = int(input("Guess x:"))
if x == 4:
 print('Correct!')
```

In this example, You guessed correctly! will only be printed if the variable x equals to four. Python can also execute multiple conditions using the elif and else keywords.

```
x = int(input("Guess x:"))
if x == 4:
 print('Correct!')
elif abs(4 - x) == 1:
 print('Wrong, but close!')
else:
 print('Wrong, way off!')
```

### 17.5.6.2 Iteration

To repeat code, the for keyword can be used. For example, to display the numbers from 1 to 10, we could write something like this:

```
for i in range(1, 11):
 print('Hello!')
```

The second argument to range, 11, is not inclusive, meaning that the loop will only get to 10 before it finishes. Python itself starts counting from 0, so this code will also work:

```
for i in range(0, 10):
 print(i + 1)
```

In fact, the range function defaults to starting value of 0, so it is equivalent to:

```
for i in range(10):
 print(i + 1)
```

We can also nest loops inside each other:

```
for i in range(0,10):
 for j in range(0,10):
 print(i, ' ',j)
```

In this case we have two nested loops. The code will iterate over the entire coordinate range (0,0) to (9,9)

## 17.5.7 Datatypes

### 17.5.7.1 Lists

see: [https://www.tutorialspoint.com/python/python\\_lists.htm](https://www.tutorialspoint.com/python/python_lists.htm)

Lists in Python are ordered sequences of elements, where each element can be accessed using a 0-based index.

To define a list, you simply list its elements between square brackets '[' ']':

```
names = [
 'Albert',
 'Jane',
 'Liz',
 'John',
 'Abby']
access the first element of the list
names[0]
'Albert'
access the third element of the list
names[2]
'Liz'
```

You can also use a negative index if you want to start counting elements from the end of the list. Thus, the last element has index -1, the second before last element has index -2 and so on:

```
access the last element of the list
names[-1]
'Abby'
access the second last element of the list
names[-2]
'John'
```

Python also allows you to take whole slices of the list by specifying a beginning and end of the slice separated by a colon

```
the middle elements, excluding first and last
names[1:-1]
['Jane', 'Liz', 'John']
```

As you can see from the example, the starting index in the slice is inclusive and the ending one, exclusive.

Python provides a variety of methods for manipulating the members of a list.

You can add elements with `append`:

```
names.append('Liz')
names
['Albert', 'Jane', 'Liz',
'John', 'Abby', 'Liz']
```

As you can see, the elements in a list need not be unique.

Merge two lists with `'extend'`:

```
names.extend(['Lindsay', 'Connor'])
```

```
names
['Albert', 'Jane', 'Liz', 'John',
'Abby', 'Liz', 'Lindsay', 'Connor']
```

Find the index of the first occurrence of an element with ‘index’:

```
names.index('Liz') \## 2
```

Remove elements by value with ‘remove’:

```
names.remove('Abby')
names
['Albert', 'Jane', 'Liz', 'John',
'Liz', 'Lindsay', 'Connor']
```

Remove elements by index with ‘pop’:

```
names.pop(1)
'Jane'
names
['Albert', 'Liz', 'John',
'Liz', 'Lindsay', 'Connor']
```

Notice that pop returns the element being removed, while remove does not.

If you are familiar with stacks from other programming languages, you can use insert and ‘pop’:

```
names.insert(0, 'Lincoln')
names
['Lincoln', 'Albert', 'Liz',
'John', 'Liz', 'Lindsay', 'Connor']
names.pop()
'Connor'
names
['Lincoln', 'Albert', 'Liz',
'John', 'Liz', 'Lindsay']
```

The Python documentation contains a [full list of list operations](#).

To go back to the range function you used earlier, it simply creates a list of numbers:

```
range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(2, 10, 2)
[2, 4, 6, 8]
```

### 17.5.7.2 Sets

Python lists can contain duplicates as you saw previously:

```
names = ['Albert', 'Jane', 'Liz',
 'John', 'Abby', 'Liz']
```

When we do not want this to be the case, we can use a [set](#):

```
unique_names = set(names)
unique_names
set(['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay'])
```

Keep in mind that the set is an unordered collection of objects, thus we can not access them by index:

```
unique_names[0]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
```

However, we can convert a set to a list easily:

```
unique_names = list(unique_names)
unique_names
['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay']
unique_names[0]
'Lincoln'
```

Notice that in this case, the order of elements in the new list matches the order in which the elements were displayed when we create the set. We had

```
set(['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay'])
```

and now we have

```
['Lincoln', 'John', 'Albert', 'Liz', 'Lindsay'])
```

You should not assume this is the case in general. That is, do not make any assumptions about the order of elements in a set when it is converted to any type of sequential data structure.

You can change a set's contents using the add, remove and update methods which correspond to the append, remove and extend methods in a list. In addition to these, set objects support the

operations you may be familiar with from mathematical sets: union, intersection, difference, as well as operations to check containment. You can read about this in the [Python documentation for sets](#).

### 17.5.7.3 Removal and Testing for Membership in Sets

One important advantage of a `set` over a `list` is that **access to elements is fast**. If you are familiar with different data structures from a Computer Science class, the Python list is implemented by an array, while the set is implemented by a hash table.

We will demonstrate this with an example. Let's say we have a list and a set of the same number of elements (approximately 100 thousand):

```
import sys, random, timeit
nums_set = set([random.randint(0, sys.maxint) for _ in range(10**5)])
nums_list = list(nums_set)
len(nums_set)
100000
```

We will use the `timeit` Python module to time 100 operations that test for the existence of a member in either the list or set:

```
timeit.timeit('random.randint(0, sys.maxint) in nums',
 setup='import random; nums=%s' % str(nums_set), number=100)
0.0004038810729980469
timeit.timeit('random.randint(0, sys.maxint) in nums',
 setup='import random; nums=%s' % str(nums_list), number=100)
0.398054122924804
```

The exact duration of the operations on your system will be different, but the take away will be the same: searching for an element in a set is orders of magnitude faster than in a list. This is important to keep in mind when you work with large amounts of data.

### 17.5.7.4 Dictionaries

One of the very important data structures in python is a dictionary also referred to as dict.

A dictionary represents a key value store:

```

person = {
 'Name': 'Albert',
 'Age': 100,
 'Class': 'Scientist'
}
print("person['Name']: ", person['Name'])
person['Name']: Albert
print("person['Age']: ", person['Age'])
person['Age']: 100

```

A convenient for to print by named attributes is

```
print("{Name} {Age}".format(**data))
```

This form of printing with the format statement and a reference to data increases readability of the print statements.

You can delete elements with the following commands:

```

del person['Name'] ## remove entry with key 'Name'
person
{'Age': 100, 'Class': 'Scientist'}
person.clear() ## remove all entries in dict
person
{}
del person ## delete entire dictionary
person
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'person' is not defined

```

You can iterate over a dict:

```

person = {
 'Name': 'Albert',
 'Age': 100,
 'Class': 'Scientist'
}
for item in person:
 print(item, person[item])

Age 100
Name Albert
Class Scientist

```

### 17.5.7.5 Dictionary Keys and Values

You can retrieve both the keys and values of a dictionary using the keys() and values() methods of the dictionary, respectively:

```
person.keys()
['Age', 'Name', 'Class']
person.values()
[100, 'Albert', 'Scientist']
```

Both methods return lists. Notice, however, that the order in which the elements appear in the returned lists (Age, Name, Class) is different from the order in which we listed the elements when we declared the dictionary initially (Name, Age, Class). It is important to keep this in mind:

- ⚠ you cannot make any assumptions about the order in which the elements of a dictionary will be returned by the `keys()` and `values()` methods\*\*.

However, you can assume that if you call `keys()` and `values()` in sequence, the order of elements will at least correspond in both methods. In the example Age corresponds to 100, Name to Albert, and Class to Scientist, and you will observe the same correspondence in general as long as `keys()` and `values()` are called one right after the other.

### 17.5.7.6 Counting with Dictionaries

One application of dictionaries that frequently comes up is counting the elements in a sequence. For example, say we have a sequence of coin flips:

```
import random
die_rolls = [
 random.choice(['heads', 'tails']) for _ in range(10)
]
die_rolls
['heads', 'tails', 'heads',
'tails', 'heads', 'heads',
'tails', 'heads', 'heads', 'heads']
```

The actual list `die_rolls` will likely be different when you execute this on your computer since the outcomes of the die rolls are random.

To compute the probabilities of heads and tails, we could count how many heads and tails we have in the list:

```
counts = {'heads': 0, 'tails': 0}
```

```

for outcome in coin_flips:
 assert outcome in counts
 counts[outcome] += 1
print('Probability of heads: %.2f' % (counts['heads'] / len(coin_flips)))
Probability of heads: 0.70

print('Probability of tails: %.2f' % (counts['tails'] / sum(counts.values())))
Probability of tails: 0.30

```

In addition to how we use the dictionary counts to count the elements of coin\_flips, notice a couple things about this example:

1. We used the assert outcome in counts statement. The assert statement in Python allows you to easily insert debugging statements in your code to help you discover errors more quickly. assert statements are executed whenever the internal Python `__debug__` variable is set to True, which is always the case unless you start Python with the -O option which allows you to run optimized Python.
2. When we computed the probability of tails, we used the built-in sum function, which allowed us to quickly find the total number of coin flips. sum is one of many built-in function you can [read about here](#).

## 17.5.8 Functions

You can reuse code by putting it inside a function that you can call in other parts of your programs. Functions are also a good way of grouping code that logically belongs together in one coherent whole. A function has a unique name in the program. Once you call a function, it will execute its body which consists of one or more lines of code:

```

def check_triangle(a, b, c):
 return \
 a < b + c and a > abs(b - c) and \
 b < a + c and b > abs(a - c) and \
 c < a + b and c > abs(a - b)

 print(check_triangle(4, 5, 6))

```

The def keyword tells Python we are defining a function. As part of

the definition, we have the function name, `check_triangle`, and the parameters of the function – variables that will be populated when the function is called.

We call the function with arguments 4, 5 and 6, which are passed in order into the parameters `a`, `b` and `c`. A function can be called several times with varying parameters. There is no limit to the number of function calls.

It is also possible to store the output of a function in a variable, so it can be reused.

```
def check_triangle(a, b, c):
 return \
 a < b + c and a > abs(b - c) and \
 b < a + c and b > abs(a - c) and \
 c < a + b and c > abs(a - b)

result = check_triangle(4, 5, 6)
print(result)
```

## 17.5.9 Classes

A class is an encapsulation of data and the processes that work on them. The data is represented in member variables, and the processes are defined in the methods of the class (methods are functions inside the class). For example, let's see how to define a `Triangle` class:

```
class Triangle(object):

 def __init__(self, length, width,
 height, angle1, angle2, angle3):
 if not self._sides_ok(length, width, height):
 print('The sides of the triangle are invalid.')
 elif not self._angles_ok(angle1, angle2, angle3):
 print('The angles of the triangle are invalid.')

 self._length = length
 self._width = width
 self._height = height

 self._angle1 = angle1
 self._angle2 = angle2
 self._angle3 = angle3

 def _sides_ok(self, a, b, c):
```

```

 return \
 a < b + c and a > abs(b - c) and \
 b < a + c and b > abs(a - c) and \
 c < a + b and c > abs(a - b)

def _angles_ok(self, a, b, c):
 return a + b + c == 180

triangle = Triangle(4, 5, 6, 35, 65, 80)

```

Python has full object-oriented programming (OOP) capabilities, however we can not cover all of them in this section, so if you need more information please refer to the [Python docs on classes and OOP](#).

## 17.5.10 Modules

Now write this simple program and save it:

```

from __future__ import print_statement, division
print("Hello world!")

```

As a check, make sure the file contains the expected contents on the command line:

```

$ cat hello.py
from __future__ import print_statement, division
print("Hello world!")

```

To execute your program pass the file as a parameter to the python command:

```

$ python hello.py
Hello world!

```

Files in which Python code is stored are called **modules**. You can execute a Python module form the command line like you just did, or you can import it in other Python code using the import statement.

Let's write a more involved Python program that will receive as input the lengths of the three sides of a triangle, and will output whether they define a valid triangle. A triangle is valid if the length of each side is less than the sum of the lengths of the other two sides and greater than the difference of the lengths of the other two sides.:.

```
"""Usage: check_triangle.py [-h] LENGTH WIDTH HEIGHT

Check if a triangle is valid.

Arguments:
LENGTH The length of the triangle.
WIDTH The width of the traingle.
HEIGHT The height of the triangle.

Options:
-h --help
"""

from __future__ import print_function, division
from docopt import docopt

if __name__ == '__main__':
 arguments = docopt(__doc__)
 a, b, c = int(arguments['LENGTH']),
 int(arguments['WIDTH']),
 int(arguments['HEIGHT']))
 valid_triangle = \
 a < b + c and a > abs(b - c) and \
 b < a + c and b > abs(a - c) and \
 c < a + b and c > abs(a - b)
 print('Triangle with sides %d, %d and %d is valid: %r' % (
 a, b, c, valid_triangle
))

```

Assuming we save the program in a file called `check_triangle.py`, we can run it like so:

```
$ python check_triangle.py 4 5 6
Triangle with sides 4, 5 and 6 is valid: True
```

Let us break this down a bit.

1. We are importing the `print_function` and `division` modules from python 3 like we did earlier in this section. It's a good idea to always include these in your programs.
2. We've defined a boolean expression that tells us if the sides that were input define a valid triangle. The result of the expression is stored in the `valid_triangle` variable. Inside are `true`, and `False` otherwise.
3. We've used the backslash symbol \ to format are code nicely. The backslash simply indicates that the current line is being continued on the next line.
4. When we run the program, we do the check if `__name__ == '__main__'`. `__name__` is an internal Python variable that allows us to

tell whether the current file is being run from the command line (value `__name__`), or is being imported by a module (the value will be the name of the module). Thus, with this statement we're just making sure the program is being run by the command line.

5. We are using the docopt module to handle command line arguments. The advantage of using this module is that it generates a usage help statement for the program and enforces command line arguments automatically. All of this is done by parsing the docstring at the top of the file.
6. In the print function, we are using [Python's string formatting capabilities](#) to insert values into the string we are displaying.

### 17.5.11 Lambda Expressions

As oppose to normal functions in Python which are defined using the `def` keyword, lambda functions in Python are anonymous functions which do not have a name and are defined using the `lambda` keyword. The generic syntax of a lambda function is in form of `lambda arguments: expression`, as shown in the following example:

```
greeter = lambda x: print('Hello %s!' %x)
print(greeter('Albert'))
```

As you could probably guess, the result is:

```
Hello Albert!
```

Now consider the following examples:

```
power2 = lambda x: x ** 2
```

The `power2` function defined in the expression, is equivalent to the following definition:

```
def power2(x):
 return x ** 2
```

Lambda functions are useful for when you need a function for a short period of time. Note that they can also be very useful when passed as

an argument with other built-in functions that take a function as an argument, e.g. `filter()` and `map()`. In the next example we show how a lambda function can be combined with the `filter` function. Consider the array `all_names` which contains five words that rhyme together. We want to filter the words that contain the word `name`. To achieve this, we pass the function `lambda x: 'name' in x` as the first argument. This lambda function returns `True` if the word `name` exists as a sub-string in the string `x`. The second argument of `filter` function is the array of names, i.e. `all_names`.

```
all_names = ['surname', 'rename', 'nickname', 'acclaims', 'defame']
filtered_names = list(filter(lambda x: 'name' in x, all_names))
print(filtered_names)
['surname', 'rename', 'nickname']
```

As you can see, the names are successfully filtered as we expected.

In Python3, `filter` function returns a `filter` object or the iterator which gets lazily evaluated which means neither we can access the elements of the `filter` object with index nor we can use `len()` to find the length of the `filter` object.

```
list_a = [1, 2, 3, 4, 5]
filter_obj = filter(lambda x: x % 2 == 0, list_a)
Convert the filter obj to a list
even_num = list(filter_obj)
print(even_num)
Output: [2, 4]
```

In Python, we can have a small usually a single liner anonymous function called Lambda function which can have any number of arguments just like a normal function but with only one expression with no return statement. The result of this expression can be applied to a value.

Basic Syntax:

```
lambda arguments : expression
```

For an example: a function in python

```
def multiply(a, b):
 return a*b
```

```
#call the function
multiply(3*5) ##outputs: 15
```

Same function can written as Lambda function. This function named as multiply is having 2 arguments and returns their multiplication.

Lambda equivalent for this function would be:

```
multiply = Lambda a, b : a*b
print(multiply(3, 5))
outputs: 15
```

Here a and b are the 2 arguments and  $a*b$  is the expression whose value is returned as an output.

Also we don't need to assign Lambda function to a variable.

```
(lambda a, b : a*b)(3*5)
```

Lambda functions are mostly passed as parameter to a function which expects a function objects like in map or filter.

### 17.5.11.1 map

The basic syntax of the map function is

```
map(function_object, iterable1, iterable2, ...)
```

map functions expects a function object and any number of iterables like list or dictionary. It executes the function\_object for each element in the sequence and returns a list of the elements modified by the function object.

Example:

```
def multiply(x):
 return x * 2

map(multiply2, [2, 4, 6, 8])
Output [4, 8, 12, 16]
```

If we want to write same function using Lambda

```
map(lambda x: x*2, [2, 4, 6, 8])
Output [4, 8, 12, 16]
```

## 17.5.11.2 dictionary

Now, lets see how we can iterate over a dictionary using map and lambda Lets say we have a dictionary object

```
dict_movies = [
 {'movie': 'avengers', 'comic': 'marvel'},
 {'movie': 'superman', 'comic': 'dc'}]
```

We can iterate over this dictionary and read the elements of it using map and lambda functions in following way:

```
map(lambda x : x['movie'], dict_movies) ## Output: ['avengers', 'superman']
map(lambda x : x['comic'], dict_movies) ## Output: ['marvel', 'dc']
map(lambda x : x['movie'] == "avengers", dict_movies)
Output: [True, False]
```

In Python3, map function returns an iterator or map object which gets lazily evaluated which means neither we can access the elements of the map object with index nor we can use len() to find the length of the map object. We can force convert the map output i.e. the map object to list as shown below:

```
map_output = map(lambda x: x*2, [1, 2, 3, 4])
print(map_output)
Output: map object: <map object at 0x04D6B80>
list_map_output = list(map_output)
print(list_map_output) ## Output: [2, 4, 6, 8]
```

## 17.5.12 Iterators

In Python, an iterator protocol is defined using two methods: `__iter__()` and `next()`. The former returns the iterator object and latter returns the next element of a sequence. Some advantages of iterators are as follows:

- Readability
- Supports sequences of infinite length
- Saving resources

There are several built-in objects in Python which implement iterator protocol, e.g. string, list, dictionary. In the following example, we create a new class that follows the iterator protocol. We then use the class to generate  $\log_2$  of numbers:

```
from math import log2

class LogTwo:
 "Implements an iterator of log two"

 def __init__(self, last = 0):
 self.last = last

 def __iter__(self):
 self.current_num = 1
 return self

 def __next__(self):
 if self.current_num <= self.last:
 result = log2(self.current_num)
 self.current_num += 1
 return result
 else:
 raise StopIteration

L = LogTwo(5)
i = iter(L)
print(next(i))
print(next(i))
print(next(i))
print(next(i))
```

As you can see, we first create an instance of the class and assign its `__iter__()` function to a variable called `i`. Then by calling the `next()` function four times, we get the following output:

```
$ python iterator.py
0.0
1.0
1.584962500721156
2.0
```

As you probably noticed, the lines are  $\log_2()$  of 1, 2, 3, 4 respectively.

## 17.5.13 Generators

Before we go to Generators, please understand Iterators. Generators are also Iterators but they can only be iterated over once. That's because Generators do not store the values in memory instead they

generate the values on the go. If we want to print those values then we can either simply iterate over them or use the for loop.

### 17.5.13.1 Generators with function

For example: we have a function named as multiplyBy10 which prints all the input numbers multiplied by 10.

```
def multiplyBy10(numbers):
 result = []
 for i in numbers:
 result.append(i*10)
 return result

new_numbers = multiplyBy10([1, 2, 3, 4, 5])

print new_numbers #Output: [10, 20, 30, 40, 50]
```

Now, if we want to use Generators here then we will make following changes.

```
def multiplyBy10(numbers):
 for i in numbers:
 yield(i*10)

new_numbers = multiplyBy10([1, 2, 3, 4, 5])

print new_numbers #Output: Generators object
```

In Generators, we use `yield()` function in place of `return()`. So when we try to print `new_numbers` list now, it just prints Generators object. The reason for this is because Generators dont hold any value in memory, it yields one result at a time. So essentially it is just waiting for us to ask for the next result. To print the next result we can just say `print next(new_numbers)`, so how it is working is its reading the first value and squaring it and yielding out value 1. Also in this case we can just print `next(new_numbers)` 5 times to print all numbers and if we do it for 6th time then we will get an error `StopIteration` which means Generators has exhausted its limit and it has no 6th element to print.

```
print next(new_numbers) #Output: 1
```

### 17.5.13.2 Generators using for loop

If we now want to print the complete list of squared values then we can just do:

```
def multiplyBy10(numbers):
 for i in numbers:
 yield(i*10)

new_numbers = multiplyBy10([1,2,3,4,5])

for num in new_numbers:
 print num
```

The output will be:

```
10
20
30
40
50
```

### 17.5.13.3 Generators with List Comprehension

Python has something called List Comprehension, if we use this then we can replace the complete function def with just:

```
new_numbers = [x*10 for x in [1,2,3,4,5]]
print new_numbers #Output: [10, 20, 30, 40 ,50]
```

Here the point to note is square brackets [] in line 1 is very important. If we change it to () then again we will start getting Generators object.

```
new_numbers = (x*10 for x in [1,2,3,4,5])
print new_numbers #Output: Generators object
```

We can get the individual elements again from Generators if we do a for loop over new\_numbers like we did previously. Alternatively, we can convert it into a list and then print it.

```
new_numbers = (x*10 for x in [1,2,3,4,5])
print list(new_numbers) #Output: [10, 20, 30, 40 ,50]
```

But here if we convert this into a list then we loose on performance, which we will just see next.

### 17.5.13.4 Why to use Generators?

Generators are better with Performance because it does not hold the values in memory and here with the small examples we provide its not a big deal since we are dealing with small amount of data but just consider a scenario where the records are in millions of data set. And if we try to convert millions of data elements into a list then that will definitely make an impact on memory and performance because everything will in memory.

Lets see an example on how Generators help in Performance. First, without Generators, normal function taking 1 million record and returns the result[people] for 1 million.

```
names = ['John', 'Jack', 'Adam', 'Steve', 'Rick']
majors = ['Math', 'CompScience', 'Arts', 'Business', 'Economics']

prints the memory before we run the function
memory = mem_profile.memory_usage_resource()
print ('Memory (Before): {memory}Mb'.format(memory=memory))

def people_list(people):
 result = []
 for i in range(people):
 person = {
 'id' : i,
 'name' : random.choice(names),
 'major' : random.choice(majors)
 }
 result.append(person)
 return result

t1 = time.clock()
people = people_list(10000000)
t2 = time.clock()

prints the memory after we run the function
memory = mem_profile.memory_usage_resource()
print ('Memory (After): {memory}Mb'.format(memory=memory))
print ('Took {time} seconds'.format(time=t2-t1))

#Output
Memory (Before): 15Mb
Memory (After): 318Mb
Took 1.2 seconds
```

I am just giving approximate values to compare it with next execution but we just try to run it we will see a serious consumption of memory with good amount of time taken.

```
names = ['John', 'Jack', 'Adam', 'Steve', 'Rick']
majors = ['Math', 'CompScience', 'Arts', 'Business', 'Economics']
```

```

prints the memory before we run the function
memory = mem_profile.memory_usage_resource()
print ('Memory (Before): {memory}Mb'.format(memory=memory))
def people_generator(people):
 for i in xrange(people):
 person = {
 'id' : i,
 'name' : random.choice(names),
 'major' : random.choice(majors)
 }
 yield person

t1 = time.clock()
people = people_list(10000000)
t2 = time.clock()

prints the memory after we run the function
memory = mem_profile.memory_usage_resource()
print ('Memory (After): {memory}Mb'.format(memory=memory))print ('Took {time} seconds'.format(
 time=t2-t1))

#Output
Memory (Before): 15Mb
Memory (After): 15Mb
Took 0.01 seconds

```

Now after running the same code using Generators, we will see a significant amount of performance boost with almost 0 Seconds. And the reason behind this is that in case of Generators, we do not keep anything in memory so system just reads 1 at a time and yields that.

## 17.5.14 Non Blocking Threads

○ Students can contribute this section

## 17.5.15 Subprocess

A module which allows us to start a new process and connect to their input, output, error nodes and get the return values is called a subprocess.

### 17.5.15.1 Popen Class

The most important class in Python to start a new process is Popen

class. The other functions like call, check\_output, and check\_call use Popen internally. Signature of this class is as follows:

```
class subprocess.Popen(args, bufsize=0, executable=None, stdin=None,
stdout=None, stderr=None, preexec_fn=None, close_fds=False, shell=False,
cwd=None, env=None, universal_newlines=False, startupinfo=None,
creationflags=0)
```

Following program starts the Unix program 'cat' and the second parameter is the argument.

```
from subprocess import Popen, PIPE

process = Popen(['cat', 'test.py'], stdout=PIPE, stderr=PIPE)
stdout, stderr = process.communicate()
print stdout
```

process.communicate() reads the input and output from the process. stderr will only get populated if there is some error. stdout is the output for this process.

### **17.5.15.2 Popen.communicate()**

The communicate() method returns a tuple (stdoutdata, stderrdata). Popen.communicate() interacts with process: Send data to stdin. Read data from stdout and stderr, until end-of-file is reached.

Wait for process to terminate.

The optional input argument should be a string to be sent to the child process, or None, if no data should be sent to the child.

Basically, when you use communicate() it means that you want to execute the command

### **17.5.15.3 Subprocess call()**

The most recommended way to launch a process is to use following function with arguments and this will also have a returncode attribute:

```
subprocess.call(args, *, stdin=None, stdout=None, stderr=None, shell=False)
Run the command described by args.
Wait for command to complete, then return the returncode attribute.
```

The behaviour of the shell argument can sometimes be confusing so I'll try to clear it a bit here.

Firstly, let's consider the case where shell is set to False, the default. In this case, if args is a string, it is assumed to be the name of the executable file. Even if it contains spaces. Consider the following.

```
subprocess.call('ls -l')
```

This won't work because subprocess is looking for an executable file called ls -l, but obviously can't find it. However, if args is a list, then the first item in this list is considered as the executable and the rest of the items in the list are passed as command line arguments to the program.

```
subprocess.call(['ls', '-l'])
```

does what you think it will.

Second case, with shell set to True, the program that actually gets executed is the OS default shell, /bin/sh on Linux and cmd.exe on windows. This can be changed with the executable argument.

When using the shell, args is usually a string, something that will be parsed by the shell program. The args string is passed as a command line argument to the shell (with a -c option on Linux) such that the shell will interpret it as a shell command sequence and process it accordingly. This means you can use all the shell builtins and goodies that your shell offers.

```
subprocess.call('ls -l', shell=True)
```

is similar to

```
$ /bin/sh -c 'ls -l'
```

In the same vein, if you pass a list as args with shell set to True, all items in the list are passed as command line arguments to the shell.

```
subprocess.call(['ls', '-l'], shell=True)
```

is similar to

```
$ /bin/sh -c ls -l
```

which is the same as

```
$ /bin/sh -c ls
```

since /bin/sh takes just the argument next to -c as the command line to execute.

Example 2:

```
>>> subprocess.call("exit 1", shell=True)
1

**Warning: Using shell=True can cause some security issues. When we have shell=True then it
If we execute shell command which takes in unsanitized input from an untrusted source, it c

>>> from subprocess import call
>>> filename = input("What file would you like to display?\n")
What file would you like to display?
non_existent; rm -rf /
>>> call("cat " + filename, shell=True) ## Uh-oh. This will end badly...

shell=False disables all shell based features, but does not suffer from this vulnerability.
```

#### 17.5.15.4 Save process output (stdout)

We can get the program output using check\_output and store it in a string which we can later print. Method definition is as follows:

```
subprocess.check_output(args, *, stdin=None, stderr=None, shell=False, universal_newlines=False,
Run command with arguments and return its output as a byte string.
```

Example:

```
>>> import subprocess
>>>
>>> s = subprocess.check_output(["echo", "Hello World!"])
>>> print("s = " + s)

'Hello World!\n'
```

If we want to get the standard error output, use stderr = subprocess.STDOUT

```
>>> subprocess.check_output("ls non_existent_file; exit 0", stderr=subprocess.STDOUT, shell=True)
'ls: non_existent_file: No such file or directory\n'
```

### 17.5.15.5 Getting the return code (OR exit status)

If we get a non-zero return code, then it will raise a CalledProcessError. This object will have return code in returncode attribute and output will be in output attribute.

```
>>> subprocess.check_output("exit 1", shell=True)
Traceback (most recent call last):
```

subprocess.CalledProcessError: Command 'exit 1' returned non-zero exit status 1

Exception subprocess.CalledProcessError

Exception raised when a process run by check\_call() or check\_output() returns a non-zero exit status.

returncode Exit status of the child process.

cmd Command that was used to spawn the child process.

output Output of the child process if this exception is raised by check\_output(). Otherwise, None.

subprocess.PIPE Special value that can be used as the stdin, stdout or stderr argument to Popen and indicates that a pipe to the standard stream should be opened. Most useful with Popen.communicate().

subprocess.STDOUT Special value that can be used as the stderr argument to Popen and indicates that standard error should go into the same handle as standard output.

```
**Note: Do not use stdout=PIPE or stderr=PIPE with this function as that can deadlock based on the platform's implementation of fork()
```

## 17.5.15.6 Popen Constructor

The process creation and its management is handled by this class - Popen. Its signature is as follows:

```
class subprocess.Popen(args, bufsize=0, executable=None, stdin=None, stdout=None, stderr=None, preexec_fn=None, shell=False, close_fds=False, universal_newlines=False, encoding=None, errors=None)
```

This will execute a child program in a new process. The arguments to Popen is as follows:

args are a sequence of program arguments or it can be a single string.

If the arguments is a sequence, then by default, the first item in args is the program to execute. If args is a string, the interpretation is platform-dependent which will see next. Unless stated specifically, it is recommended to pass args as a sequence.

On Unix, if args is a string, the string is interpreted as the name or path of the program to execute. However, this can only be done if not passing arguments to the program.

```
Note shlex.split() can be useful when determining the correct tokenization for args, especially when options like -input and -output are present.
>>> import shlex, subprocess
>>> command_line = raw_input()
/bin/vikings -input eggs.txt -output "spam spam.txt" -cmd "echo '$MONEY'"
>>> args = shlex.split(command_line)
>>> print args
['/bin/vikings', '-input', 'eggs.txt', '-output', 'spam spam.txt', '-cmd', "echo '$MONEY'"]
>>> p = subprocess.Popen(args) ## Success!
```

Note in particular that options (such as -input) and arguments (such as eggs.txt) that are

On Windows, if args is a sequence then it will be converted to a string. This is because the underlying CreateProcess() operates on strings. Parsing the string after conversion uses the following rules:

1. Arguments are delimited by white space, which is either a space or a tab.
2. A string surrounded by double quotation marks is interpreted as a single argument, regardless of white space contained within. A quoted string can be embedded in an argument.
3. A double quotation mark preceded by a backslash is

- interpreted as a literal double quotation mark.
4. Backslashes are interpreted literally, unless they immediately precede a double quotation mark.
  5. If backslashes immediately precede a double quotation mark, every pair of backslashes is interpreted as a literal backslash. If the number of backslashes is odd, the last backslash escapes the next double quotation mark as described in rule 3.

The shell argument is by default set to False, this argument specifies whether to use the shell as the program to execute. If shell is True, it is recommended to pass args as a string rather than as a sequence.

#### **17.5.15.7 Exceptions in Subprocess**

If a child process raises any exception before the new program starts, that exception will be raised again in the parent process. Additionally, the exception object will have one extra attribute called child\_traceback, which is a string containing traceback information from the child's point of view.

OSError - This occurs, for example, when trying to execute a non-existent file. Applications should prepare for OSError exceptions.

ValueError - This will be raised if Popen is called with invalid arguments.

CalledProcessError - check\_call() and check\_output() will raise CalledProcessError if the called process returns a non-zero return code.

#### **17.5.15.8 Security**

It's very important for the application to handle security aspect explicitly.

#### **17.5.15.9 Popen Objects**

Popen.poll() Check if child process has terminated. Set and return returncode attribute.

Popen.wait() Wait for child process to terminate. Set and return returncode attribute.

```
**Warning This will deadlock when using stdout=PIPE and/or stderr=PIPE and the child process's output is never read.
```

Popen.communicate(input=None) Interact with process: Send data to stdin. Read data from stdout and stderr, until end-of-file is reached. Wait for process to terminate. The optional input argument should be a string to be sent to the child process, or None, if no data should be sent to the child. communicate() returns a tuple (stdoutdata, stderrdata).

```
**Note that if you want to send data to the process's stdin, you need to create the Popen object with shell=True.
```

```
**Note The data read is buffered in memory, so do not use this method if the data size is large.
```

Popen.send\_signal(signal) Sends the signal signal to the child.

```
**Note On Windows, SIGTERM is an alias for terminate(). CTRL_C_EVENT and CTRL_BREAK_EVENT are aliases for SIGINT and SIGBREAK.
```

New in version 2.6.

Popen.terminate() Stop the child. On Posix OSs the method sends SIGTERM to the child. On Windows the Win32 API function TerminateProcess() is called to stop the child.

New in version 2.6.

Popen.kill() Kills the child. On Posix OSs the function sends SIGKILL to the child. On Windows kill() is an alias for terminate().

New in version 2.6.

The following attributes are also available:

```
**Warning Use communicate() rather than .stdin.write, .stdout.read or .stderr.read to avoid deadlocks if the child process takes a long time to read from its input or write to its output.
```

Popen.stdin If the stdin argument was PIPE, this attribute is a file object that provides input to the child process. Otherwise, it is None.

Popen.stdout If the stdout argument was PIPE, this attribute is a file object that provides output from the child process. Otherwise, it is None.

Popen.stderr If the stderr argument was PIPE, this attribute is a file object that provides error output from the child process. Otherwise, it is None.

Popen.pid The process ID of the child process.

\*\*Note that if you set the shell argument to True, this is the process ID of the spawned shell, not the child process.

Popen.returncode The child return code, set by poll() and wait() (and indirectly by communicate()). A None value indicates that the process hasn't terminated yet.

A negative value -N indicates that the child was terminated by signal N (Unix only).

### 17.5.16 Queue

○ Students can contribute this section

see: \* <https://docs.python.org/3/library/queue.html>

### 17.5.17 Scheduler

○ Students can contribute this section

see: \* <https://docs.python.org/3/library/sched.html>

### 17.5.18 Python SSL

○ Students can contribute this section

see:

- <https://docs.python.org/3/library/ssl.html>
- also demonstrate how you could just use subprocess ... to contrast

## 17.6 PYTHON MODULES

---



Often you may need functionality that is not present in Python's standard library. In this case you have two options:

- implement the features yourself
- use a third-party library that has the desired features.

Often you can find a previous implementation of what you need. Since this is a common situation, there is a service supporting it: the [Python Package Index](#) (or PyPi for short).

Our task here is to install the [autopep8](#) tool from PyPi. This will allow us to illustrate the use of virtual environments using the pyenv or virtualenv command, and installing and uninstalling PyPi packages using pip.

### 17.6.1 Updating Pip

It is important that you have the newest version of pip installed for your version of python. Let us assume your python is registered with pyenv and you use pyenv, then you can update pip with

```
pip install -U pip
```

without interfering with a potential system wide installed version of pip that may be needed by the system default version of python. See the section about pyenv for more details

### 17.6.2 Using pip to Install Packages

Let's now look at another important tool for Python development: the

Python Package Index, or PyPI for short. PyPI provides a large set of third-party python packages. If you want to do something in python, first check pypi, as odd are someone already ran into the problem and created a package solving it.

In order to install package from PyPI, use the pip command. We can search for PyPI for packages:

```
$ pip search --trusted-host pypi.python.org autopep8 pylint
```

It appears that the top two results are what we want so install them:

```
$ pip install --trusted-host pypi.python.org autopep8 pylint
```

This will cause pip to download the packages from PyPI, extract them, check their dependencies and install those as needed, then install the requested packages.

You can skip ‘–trusted-host pypi.python.org’ option if you have patched urllib3 on Python 2.7.9.

## 17.6.3 GUI

### 17.6.3.1 GUIZero

Install guizero with the following command:

```
sudo pip install guizero
```

For a comprehensive tutorial on guizero, [click here](#).

### 17.6.3.2 Kivy

You can install Kivy on macOS as follows:

```
brew install pkg-config sdl2 sdl2_image sdl2_ttf sdl2_mixer gstreamer
pip install -U Cython
pip install kivy
pip install pygame
```

A hello world program for kivy is included in the cloudmesh.robot repository. Which you can find here

- <https://github.com/cloudmesh/cloudmesh.robot/tree/master/>

To run the program, please download it or execute it in cloudmesh.robot as follows:

```
cd cloudmesh.robot/projects/kivy
python swim.py
```

To create stand alone packages with kivy, please see:

```
- https://kivy.org/docs/guide/packaging-osx.html
```

## 17.6.4 Formatting and Checking Python Code

First, get the bad code:

```
$ wget --no-check-certificate http://git.io/pXqb -O bad_code_example.py
```

Examine the code:

```
$ emacs bad_code_example.py
```

As you can see, this is very dense and hard to read. Cleaning it up by hand would be a time-consuming and error-prone process. Luckily, this is a common problem so there exist a couple packages to help in this situation.

## 17.6.5 Using autopep8

We can now run the bad code through autopep8 to fix formatting problems:

```
$ autopep8 bad_code_example.py >code_example_autopep8.py
```

Let us look at the result. This is considerably better than before. It is easy to tell what the example1 and example2 functions are doing.

It is a good idea to develop a habit of using autopep8 in your python-

development workflow. For instance: use autopep8 to check a file, and if it passes, make any changes in place using the -i flag:

```
$ autopep8 file.py # check output to see of passes
$ autopep8 -i file.py # update in place
```

If you use pyCharm you have the ability to use a similar function while pressing on Inspect Code.

## 17.6.6 Writing Python 3 Compatible Code

To write python 2 and 3 compatible code we recommend that you take a look at: [http://python-future.org/compatible\\_idioms.html](http://python-future.org/compatible_idioms.html)

## 17.6.7 Using Python on FutureSystems

This is only important if you use Futuresystems resources.

In order to use Python you must log into your FutureSystems account. Then at the shell prompt execute the following command:

```
$ module load python
```

This will make the python and virtualenv commands available to you.

The details of what the module load command does are described in the future lesson modules.

## 17.6.8 Ecosystem

### 17.6.8.1 pypi

The Python Package Index is a large repository of software for the Python programming language containing a large number of packages, many of which can be found on [pypi](#). The nice thing about pypi is that many packages can be installed with the program 'pip'.

To do so you have to locate the <package\_name> for example with the search function in pypi and say on the commandline:

```
$ pip install <package_name>
```

where `package_name` is the string name of the package. an example would be the package called `cloudmesh_client` which you can install with:

```
$ pip install cloudmesh_client
```

If all goes well the package will be installed.

### 17.6.8.2 Alternative Installations

The basic installation of python is provided by [python.org](http://python.org). However others claim to have alternative environments that allow you to install python. This includes

- [Canopy](#)
- [Anaconda](#)
- [IronPython](#)

Typically they include not only the python compiler but also several useful packages. It is fine to use such environments for the class, but it should be noted that in both cases not every python library may be available for install in the given environment. For example if you need to use `cloudmesh client`, it may not be available as `conda` or `Canopy` package. This is also the case for many other cloud related and useful python libraries. Hence, we do recommend that if you are new to python to use the distribution from [python.org](http://python.org), and use `pip` and `virtualenv`.

Additionally some python version have platform specific libraries or dependencies. For example coca libraries, `.NET` or other frameworks are examples. For the assignments and the projects such platform dependent libraries are not to be used.

If however you can write a platform independent code that works on Linux, macOS and Windows while using the [python.org](http://python.org) version but develop it with any of the other tools that is just fine. However it is up to you to guarantee that this independence is maintained and implemented. You do have to write `requirements.txt` files that will

install the necessary python libraries in a platform independent fashion. The homework assignment PRG1 has even a requirement to do so.

In order to provide platform independence we have given in the class a minimal python version that we have tested with hundreds of students: python.org. If you use any other version, that is your decision. Additionally some students not only use python.org but have used iPython which is fine too. However this class is not only about python, but also about how to have your code run on any platform. The homework is designed so that you can identify a setup that works for you.

However we have concerns if you for example wanted to use chameleon cloud which we require you to access with cloudmesh. cloudmesh is not available as conda, canopy, or other framework package. Cloudmesh client is available from pypi which is standard and should be supported by the frameworks. We have not tested cloudmesh on any other python version than python.org which is the open source community standard. None of the other versions are standard.

In fact we had students over the summer using canopy on their machines and they got confused as they now had multiple python versions and did not know how to switch between them and activate the correct version. Certainly if you know how to do that, than feel free to use canopy, and if you want to use canopy all this is up to you. However the homework and project requires you to make your program portable to python.org. If you know how to do that even if you use canopy, anaconda, or any other python version that is fine. Graders will test your programs on a python.org installation and not canopy, anaconda, ironpython while using virtualenv. It is obvious why. If you do not know that answer you may want to think about that every time they test a program they need to do a new virtualenv and run vanilla python in it. If we were to run two installs in the same system, this will not work as we do not know if one student will cause a side effect for another. Thus we as instructors do not just have to

look at your code but code of hundreds of students with different setups. This is a non scalable solution as every time we test out code from a student we would have to wipe out the OS, install it new, install a new version of whatever python you have elected, become familiar with that version and so on and on. This is the reason why the open source community is using python.org. We follow best practices. Using other versions is not a community best practice, but may work for an individual.

We have however in regards to using other python version additional bonus projects such as

- deploy run and document cloudmesh on ironpython
- deploy run and document cloudmesh on anaconda, develop script to generate a conda package from github
- deploy run and document cloudmesh on canopy, develop script to generate a conda package from github
- deploy run and document cloudmesh on ironpython
- other documentation that would be useful

## 17.6.9 Resources

If you are unfamiliar with programming in Python, we also refer you to some of the numerous online resources. You may wish to start with [Learn Python](#) or the book [Learn Python the Hard Way](#). Other options include [Tutorials Point](#) or [Code Academy](#), and the Python wiki page contains a long list of [references for learning](#) as well. Additional resources include:

- <https://virtualenvwrapper.readthedocs.io>
- <https://github.com/yyuu/pyenv>
- <https://amaral.northwestern.edu/resources/guides/pyenv-tutorial>
- <https://godjango.com/96-django-and-python-3-how-to-setup-pyenv-for-multiple-pythons/>
- <https://www.accelebrate.com/blog/the-many-faces-of-python-and-how-to-manage-them/>

- <http://ivory.idyll.org/articles/advanced-swc/>
- <http://python.net/~goodger/projects/pycon/2007/idiomatic/handouts/idiomatic.html>
- <http://www.youtube.com/watch?v=0vJlVBVTFg>
- <http://www.korokithakis.net/tutorials/python/>
- <http://www.afterhoursprogramming.com/tutorial/Python/Intro/>
- <http://www.greenteapress.com/thinkpython/thinkCSpypdf.pdf>
- <https://docs.python.org/3.3/tutorial/modules.html>
- [https://www.learnpython.org/en/Modules\\_and\\_Packages](https://www.learnpython.org/en/Modules_and_Packages)
- <https://docs.python.org/2/library/datetime.html>
- [https://chrisalbon.com/python/strings\\_to\\_datetime.html](https://chrisalbon.com/python/strings_to_datetime.html)

A very long list of useful information are also available from

- <https://github.com/vinta/awesome-python>
- [https://github.com/rasbt/python\\_reference](https://github.com/rasbt/python_reference)

This list may be useful as it also contains links to data visualization and manipulation libraries, and AI tools and libraries. Please note that for this class you can reuse such libraries if not otherwise stated.

### **17.6.9.1 Jupyter Notebook Tutorials**

A Short Introduction to Jupyter Notebooks and NumPy To view the notebook, open this link in a background tab <https://nbviewer.jupyter.org/> and copy and paste the following link in the URL input area <https://cloudmesh.github.io/classes/lesson/prg/Jupyter-NumPy-tutorial-I523-F2017.ipynb> Then hit Go.

### **17.6.10 Exercises**

E.Python.Lib.1:

Write a python program called iterate.py that accepts an integer n from the command line. Pass this integer to a function called iterate.

The iterate function should then iterate from 1 to n. If the i-th number is a multiple of three, print multiple of 3, if a multiple of 5 print multiple of 5, if a multiple of both print multiple of 3 and 5, else print the value.

E:Python.Lib.2:

1. Create a pyenv or virtualenv ~/ENV
2. Modify your ~/.bashrc shell file to activate your environment upon login.
3. Install the docopt python package using pip
4. Write a program that uses docopt to define a commandline program. Hint: modify the iterate program.
5. Demonstrate the program works.

## 17.6.11 DATA MANAGEMENT



Obviously when dealing with big data we may not only be dealing with data in one format but in many different formats. It is important that you will be able to master such formats and seamlessly integrate in your analysis. Thus we provide some simple examples on which different data formats exist and how to use them.

### 17.6.11.1 Formats

#### 17.6.11.1.1 Pickle

Python pickle allows you to save data in a python native format into a file that can later be read in by other programs. However, the data format may not be portable among different python versions thus the format is often not suitable to store information. Instead we recommend for standard data to use either json or yaml.

```
import pickle

flavor = {
 "small": 100,
 "medium": 1000,
 "large": 10000
}

pickle.dump(flavor, open("data.p", "wb"))
```

To read it back in use

```
flavor = pickle.load(open("data.p", "rb"))
```

#### 17.6.11.1.2 Text Files

To read text files into a variable called content you can use

```
content = open('filename.txt', 'r').read()
```

You can also use the following code while using the convenient `with` statement

```
with open('filename.txt', 'r') as file:
 content = file.read()
```

To split up the lines of the file into an array you can do

```
with open('filename.txt', 'r') as file:
 lines = file.read().splitlines()
```

This can also be done with the build in `readlines` function

```
lines = open('filename.txt', 'r').readlines()
```

In case the file is too big you will want to read the file line by line:

```
with open('filename.txt', 'r') as file:
 line = file.readline()
 print (line)
```

#### 17.6.11.1.3 CSV Files

Often data is contained in comma separated values (CSV) within a file. To read such files you can use the `csv` package.

```
import csv
```

```
with open('data.csv', 'rb') as f:
 contents = csv.reader(f)
for row in content:
 print row
```

Using pandas you can read them as follows.

```
import pandas as pd
df = pd.read_csv("example.csv")
```

There are many other modules and libraries that include CSV read functions. IN case you need to split a single line by comma, you may also use the `split` function. However, remember it will split at every comma, including those contained in quotes. SO this method although looking originally convenient has limitations.

#### 17.6.11.1.4 Excel spread sheets

Pandas contains a method to read Excel files

```
import pandas as pd
filename = 'data.xlsx'
data = pd.ExcelFile(file)
df = data.parse('Sheet1')
```

#### 17.6.11.1.5 YAML

YAML is a very important format as it allows you easily to structure data in hierarchical fields It is frequently used to coordinate programs while using yaml as the specification for configuration files, but also data files. To read in a yaml file the following code can be used

```
import yaml
with open('data.yaml', 'r') as f:
 content = yaml.load(f)
```

The nice part is that this code can also be used to verify if a file is valid yaml. To write data out we can use

```
with open('data.yml', 'w') as f:
 yaml.dump(data, f, default_flow_style=False)
```

The flow style set to false formats the data in a nice readable fashion with indentations.

#### 17.6.11.1.6 JSON

```
import json
with open('strings.json') as f:
 content = json.load(f)
```

#### 17.6.11.1.7 XML

Please contribute a XML python section

#### 17.6.11.1.8 RDF

To read RDF files you will need to install RDFlib with

```
$ pip install rdflib
```

This will than allow you to read RDF files

```
from rdflib.graph import Graph
g = Graph()
g.parse("filename.rdf", format="format")
for entry in g:
 print(entry)
```

Good examples on using RDF are provided on the RDFlib Web page at  
<https://github.com/RDFLib/rdflib>

From the Web page we showcase also how to directly process RDF data from the Web

```
import rdflib
g=rdflib.Graph()
g.load('http://dbpedia.org/resource/Semantic_Web')

for s,p,o in g:
 print s,p,o
```

#### 17.6.11.1.9 PDF

The Portable Document Format (PDF) has been made available by Adobe Inc. royalty free. This has enabled PDF to become a world wide adopted format that also has been standardized in 2008 (ISO/IEC 32000-1:2008, <https://www.iso.org/standard/51502.html>). A lot of research is published in papers making PDF one of the de-facto

standards for publishing. However, PDF is difficult to parse and is focused on high quality output instead of data representation. Nevertheless, tools to manipulate PDF exist:

### PDFMiner

<https://pypi.python.org/pypi/pdfminer/> allows the simple translation of PDF into text that can be further mined. The manual page helps to demonstrate some examples <http://euske.github.io/pdfminer/index.html>.

### pdf-parser.py

<https://blog.didierstevens.com/programs/pdf-tools/> parses pdf documents and identifies some structural elements that can than be further processed.

If you know about other tools, let us know.

#### 17.6.11.1.10 HTML

A very powerful library to parse HTML Web pages is provided with <https://www.crummy.com/software/BeautifulSoup/>

More details about it are provided in the documentation page <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

○ TODO: Students can contribute a section

#### 17.6.11.1.11 ConfigParser

○ TODO: Students can contribute a section

- <https://pymotw.com/2/ConfigParser/>

#### 17.6.11.1.12 ConfigDict

- <https://github.com/cloudmesh/cloudmesh.common/blob/master/cloudmesh/common/configdict.py>

## 17.6.11.2 Encryption

Often we need to protect the information stored in a file. This is achieved with encryption. There are many methods of supporting encryption and even if a file is encrypted it may be target to attacks. Thus it is not only important to encrypt data that you do not want others to see but also to make sure that the system on which the data is hosted is secure. This is especially important if we talk about big data having a potential large effect if it gets into the wrong hands.

To illustrate one type of encryption that is non trivial we have chosen to demonstrate how to encrypt a file with an ssh key. In case you have openssl installed on your system, this can be achieved as follows.

```
#!/bin/sh

Step 1. Creating a file with data
echo "Big Data is the future." > file.txt

Step 2. Create the pem
openssl rsa -in ~/.ssh/id_rsa -pubout > ~/.ssh/id_rsa.pub.pem

Step 3. look at the pem file to illustrate how it looks like (optional)
cat ~/.ssh/id_rsa.pub.pem

Step 4. encrypt the file into secret.txt
openssl rsautl -encrypt -pubin -inkey ~/.ssh/id_rsa.pub.pem -in file.txt -out secret.txt

Step 5. decrypt the file and print the contents to stdout
openssl rsautl -decrypt -inkey ~/.ssh/id_rsa -in secret.txt
```

Most important here are Step 4 that encrypts the file and Step 5 that decrypts the file. Using the Python os module it is straight forward to implement this. However, we are providing in cloudmesh a convenient class that makes the use in python very simple.

```
from cloudmesh.common.ssh.encrypt import EncryptFile

e = EncryptFile('file.txt', 'secret.txt')
e.encrypt()
e.decrypt()
```

In our class we initialize it with the locations of the file that is to be

encrypted and decrypted. To initiate that action just call the methods `encrypt` and `decrypt`.

### 17.6.11.3 Database Access

○ TODO: Students: define conventional database access section

see:

[https://www.tutorialspoint.com/python/python\\_database\\_access.htm](https://www.tutorialspoint.com/python/python_database_access.htm)

### 17.6.11.4 SQLite

;o: TODO: Students can contribute to this section

<https://www.sqlite.org/index.html>

<https://docs.python.org/3/library/sqlite3.html>

#### 17.6.11.4.1 Exercises ○

E:Encryption.1:

Test the shell script to replicate how this example works

E:Encryption.2:

Test the cloudmesh encryption class

E:Encryption.3:

What other encryption methods exist. Can you provide an example and contribute to the section?

E:Encryption.4:

What is the issue of encryption that make it challenging for Big Data

E:Encryption.5:

Given a test dataset with many files text files, how long will it take to encrypt and decrypt them on various machines. Write a benchmark that you test. Develop this benchmark as a group, test out the time it takes to execute it on a variety of platforms.

## 17.6.12 PLOTTING WITH MATPLOTLIB



A brief overview of plotting with matplotlib along with examples is provided. First matplotlib must be installed, which can be accomplished with pip install as follows:

```
pip install matplotlib
```

We will start by plotting a simple line graph using built in numpy functions for sine and cosine. This first step is to import the proper libraries shown below.

```
import numpy as np
import matplotlib.pyplot as plt
```

Next we will define the values for the x axis, we do this with the linspace option in numpy. The first two parameters are the starting and ending points, these must be scalars. The third parameter is optional and defines the number of samples to be generated between the starting and ending points, this value must be an integer. Additional parameters for the linspace utility can be found here:

```
x = np.linspace(-np.pi, np.pi, 16)
```

Now we will use the sine and cosine functions in order to generate y values, for this we will use the values of x for the argument of both our sine and cosine functions i.e.  $\cos(x)$ .

```
cos = np.cos(x)
sin = np.sin(x)
```

You can display the values of the three parameters we have defined

by typing them in a python shell.

```
x
array([-3.14159265, -2.72271363, -2.30383461, -1.88495559, -1.46607657,
 -1.04719755, -0.62831853, -0.20943951, 0.20943951, 0.62831853,
 1.04719755, 1.46607657, 1.88495559, 2.30383461, 2.72271363,
 3.14159265])
```

Having defined x and y values we can generate a line plot and since we imported matplotlib.pyplot as plt we simply use plt.plot.

```
plt.plot(x,cos)
```

We can display the plot using plt.show() which will pop up a figure displaying the plot defined.

```
plt.show()
```

Additionally we can add the sine line to our line graph by entering the following.

```
plt.plot(x,sin)
```

Invoking plt.show() now will show a figure with both sine and cosine lines displayed. Now that we have a figure generated it would be useful to label the x and y axis and provide a title. This is done by the following three commands below:

```
plt.xlabel("X - label (units)")
plt.ylabel("Y - label (units)")
plt.title("A clever Title for your Figure")
```

Along with axis labels and a title another useful figure feature may be a legend. In order to create a legend you must first designate a label for the line, this label will be what shows up in the legend. The label is defined in the initial plt.plot(x,y) instance, below is an example.

```
plt.plot(x,cos, label="cosine")
```

Then in order to display the legend the following command is issued:

```
plt.legend(loc='upper right')
```

The location is specified by using upper or lower and left or right.

Naturally all these commands can be combined and put in a file with the .py extension and run from the command line.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-np.pi, np.pi, 16)
cos = np.cos(x)
sin = np.sin(x)
plt.plot(x,cos, label="cosine")
plt.plot(x,sin, label="sine")

plt.xlabel("X - label (units)")
plt.ylabel("Y - label (units)")
plt.title("A clever Title for your Figure")

plt.legend(loc='upper right')

plt.show()
```

## ○ link error

An example of a bar chart is preceded below using data from [\[T:fast-cars\]](#).

```
import matplotlib.pyplot as plt

x = ['Toyota Prius', 'Tesla Roadster ', 'Bugatti Veyron', 'Honda Civic ', 'Lamborghini
horse_power = [120, 288, 1200, 158, 695]

x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, horse_power, color='green')
plt.xlabel("Car Model")
plt.ylabel("Horse Power (Hp)")
plt.title("Horse Power for Selected Cars")

plt.xticks(x_pos, x)

plt.show()
```

You can customize plots further by using plt.style.use(), in python 3. If you provide the following command inside a python command shell you will see a list of available styles.

```
print(plt.style.available)
```

An example of using a predefined style is shown below.

```
plt.style.use('seaborn')
```

Up to this point we have only showcased how to display figures through python output, however web browsers are a popular way to display figures. One example is Bokeh, the following lines can be entered in a python shell and the figure is outputted to a browser.

```
from bokeh.io import show
from bokeh.plotting import figure

x_values = [1, 2, 3, 4, 5]
y_values = [6, 7, 2, 3, 6]

p = figure()
p.circle(x=x_values, y=y_values)
show(p)
```

## 17.6.13 DocOPTS



When we want to design commandline arguments for python programs we have many options. However, as our approach is to create documentation first, docopts provides also a good approach for Python. The code for it is located at

- <https://github.com/docopt/docopt>

It can be installed with

```
$ pip install docopt
```

A sample programs are located at

- <https://github.com/docopt/docopt/blob/master/examples/opt>

A sample program of using doc opts for our purposes looks as follows

```
"""Cloudmesh VM management

Usage:
cm-go vm start NAME [--cloud=CLOUD]
cm-go vm stop NAME [--cloud=CLOUD]
cm-go set --cloud=CLOUD
cm-go -h | --help
cm-go --version

Options:
```

```

-h --help Show this screen.
--version Show version.
--cloud=CLOUD The name of the cloud.
--moored Moored (anchored) mine.
--drifting Drifting mine.

ARGUMENTS:
NAME The name of the VM
"""

from docopt import docopt

if __name__ == '__main__':
 arguments = docopt(__doc__, version='1.0.0rc2')
 print(arguments)

```

Another good feature of using docopts is that we can use the same verbal description in other programming languages as showcased in this book.

## 17.6.14 CLOUDMESH COMMAND SHELL



### 17.6.14.1 CMD5

Python's CMD (<https://docs.python.org/2/library/cmd.html>) is a very useful package to create command line shells. However it does not allow the dynamic integration of newly defined commands. Furthermore, additions to CMD need to be done within the same source tree. To simplify developing commands by a number of people and to have a dynamic plugin mechanism, we developed cmd5. It is a rewrite on our earlier efforts in cloudmesh client and cmd3.

#### 17.6.14.1.1 Resources

The source code for cmd5 is located in github:

- <https://github.com/cloudmesh/cmd5>

#### 17.6.14.1.2 Creating a Python Development Environment

We recommend that you use a virtualenv either with virtualenv or pyenv. This is in detail documented in the Section [\[S:managing-](#)

[multiple-python-versions-with-pyenv\].](#)

#### **17.6.14.1.3 Installation from source**

Cmd5 can be easily deployed with pip:

```
$ pip install cloudmesh.cmd5
```

In case you would like to generate easily new cmd5 commands we also recommend you install the cloudmesh sys command with:

```
$ pip install cloudmesh.sys
```

In case you like to work with the source please clone the following directories from github:

```
mkdir -p ~/github
cd ~/github

git clone https://github.com/cloudmesh/cloudmesh.common.git
git clone https://github.com/cloudmesh/cloudmesh.cmd5.git
git clone https://github.com/cloudmesh/cloudmesh.sys.git

cd ~/github/cloudmesh.common
python setup.py install
pip install .

cd ~/github/cloudmesh.cmd5
python setup.py install
pip install .

cd ~/github/cloudmesh.sys
python setup.py install
pip install .
```

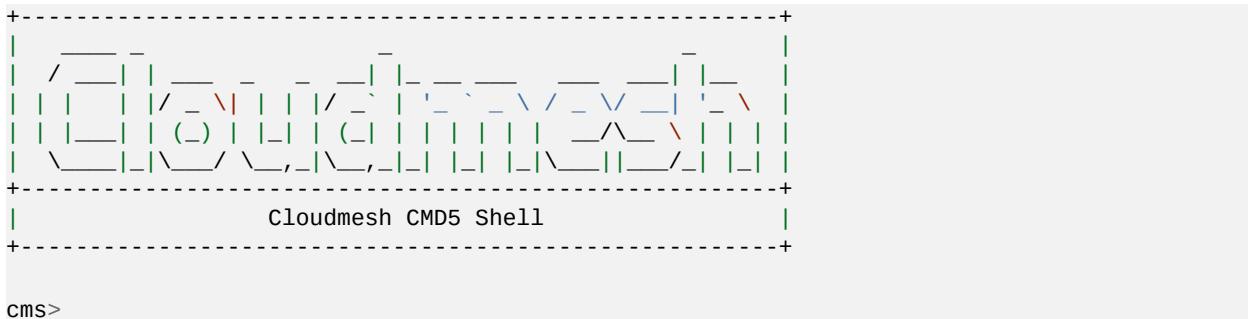
The common directory contains some useful libraries, the cmd5 repository >contains the shell, while the sys directory contains a command to generate extensions to cloudmesh.

#### **17.6.14.1.4 Execution**

To run the shell you can activate it with the cms command. cms stands for cloudmesh shell:

```
(ENV2) $ cms
```

It will print the banner and enter the shell:



To see the list of commands you can say:

cms> help

To see the manual page for a specific command, please use:

help COMMANDNAME

### **17.6.14.1.5 Create your own Extension**

One of the most important features of CMD5 is its ability to extend it with new commands. This is done via packaged name spaces. We recommend you name is cloudmesh.mycommand, where mycommand is the name of the command that you like to create. This can easily be done while using the sys command:

```
$ cms sys command generate mycommand
```

It will download a template from cloudmesh called `cloudmesh.bar` and generate a new directory `cloudmesh.mycommand` with all the needed files to create your own command and register it dynamically with cloudmesh. All you have to do is to cd into the directory and install the code:

```
$ cd cloudmesh.mycommand
$ python setup.py install
pip install .
```

Adding your own command is easy. It is important that all objects are defined in the command itself and that no global variables be used in order to allow each shell command to stand alone. Naturally you

should develop API libraries outside of the cloudmesh shell command and reuse them in order to keep the command code as small as possible. We place the command in:

```
cloudmesh/mycommand/command/mycommand.py
```

An example for the bar command is presented at:

- <https://github.com/cloudmesh/cloudmesh.bar/blob/master/cloudmesh/mycommand/command/bar.py>

It shows how simple the command definition is (bar.py):

```
from __future__ import print_function
from cloudmesh.shell.command import command
from cloudmesh.shell.command import PluginCommand

class BarCommand(PluginCommand):

 @command
 def do_bar(self, args, arguments):
 """
 ::

 Usage:
 command -f FILE
 command FILE
 command list
 This command does some useful things.
 Arguments:
 FILE a file name
 Options:
 -f specify the file
 """
 print(arguments)
```

An important difference to other CMD solutions is that our commands can leverage (besides the standard definition), docopts as a way to define the manual page. This allows us to use arguments as dict and use simple if conditions to interpret the command. Using docopts has the advantage that contributors are forced to think about the command and its options and document them from the start. Previously we did not use but argparse and click. However we noticed that for our contributors both systems lead to commands that were either not properly documented or the developers delivered ambiguous commands that resulted in confusion and wrong usage by subsequent users. Hence, we do recommend that you use docopts for documenting cmd5 commands. The

transformation is enabled by the `@command` decorator that generates a manual page and creates a proper help message for the shell automatically. Thus there is no need to introduce a separate help method as would normally be needed in CMD while reducing the effort it takes to contribute new commands in a dynamic fashion.

#### **17.6.14.1.6 Exercises**

##### E.CMD5.1

Install cmd5 on your computer.

##### E.CMD5.2

Write a new command with your firstname as the command name.

##### E.CMD5.3

Write a new command and experiment with docopt syntax and argument interpretation of the dict with if conditions.

##### E.CMD5.4

If you have useful extensions that you like us to add by default, please work with us.

##### E.CMD5.5

At this time one needs to quote in some commands the `"` in the shell command line. Develop and test code that fixes this.

#### **17.6.15 cmd MODULE**



If you consider using this module, you may instead want to use cloudmesh cmd5 instead as it provides some very nice features that are not included in cmd. However to do the basics, cmd will do.

The Python cmd module is useful for any more involved command-line application. It is used in the [Cloudmesh Project](#), for example, and students have found it helpful in their projects to develop quickly high quality command line tools with documentation so that others can replicate and use the programs. The Python cmd module contains a public class, Cmd, designed to be used as a base class for command processors such as interactive shells and other command interpreters.

### 17.6.15.1 Hello, World with cmd

This example shows a very simple command interpreter that simply responds to the greet command.

In order to demonstrate commands provided by cmd, let's save the following program in a file called helloworld.py.

```
from __future__ import print_function, division
import cmd

class HelloWorld(cmd.Cmd):
 '''Simple command processor example.'''

 def do_greet(self, line):
 if line is not None and len(line.strip()) > 0:
 print('Hello, %s!' % line.strip().title())
 else:
 print('Hello!')

 def do_EOF(self, line):
 print('bye, bye')
 return True

if __name__ == '__main__':
 HelloWorld().cmdloop()
```

A session with this program might look like this:

```
$ python helloworld.py

(Cmd) help

Documented commands (type help <topic>):
=====

```

```
help

Undocumented commands:
=====
EOF greet

(Cmd) greet
Hello!
(Cmd) greet albert
Hello, Albert!
<CTRL-D pressed>
(Cmd) bye, bye
```

The Cmd class can be used to customize a subclass that becomes a user-defined command prompt. After you have executed your program, commands defined in your class can be used. Take note of the following in this example:

- The methods of the class of the form do\_xxx implement the shell commands, with xxx being the name of the command. For example, in the `HelloWorld` class, the function `do_greet` maps to the `greet` on the command line.
- The EOF command is a special command that is executed when you press CTRL-D on your keyboard.
- As soon as any command method returns True the shell application exits. Thus, in this example the shell is exited by pressing CTRL-D, since the `do_EOF` method is the only one that returns True.
- The shell application is started by calling the `cmdloop` method of the class.

### 17.6.15.2 A More Involved Example

Let's look at a little more involved example. Save the following code in a file called `calculator.py`.

```
from __future__ import print_function, division
import cmd

class Calculator(cmd.Cmd):
```

```

prompt = 'calc >>> '
intro = 'Simple calculator that can do addition, subtraction, multiplication and division.'

def do_add(self, line):
 args = line.split()
 total = 0
 for arg in args:
 total += float(arg.strip())
 print(total)

def do_subtract(self, line):
 args = line.split()
 total = 0
 if len(args) > 0:
 total = float(args[0])
 for arg in args[1:]:
 total -= float(arg.strip())
 print(total)

def do_EOF(self, line):
 print('bye, bye')
 return True

if __name__ == '__main__':
 Calculator().cmdloop()

```

A session with this program might look like this:

```

$ python calculator.py
Simple calculator that can do addition, subtraction, multiplication and division.
calc >>> help

Documented commands (type help <topic>):
=====
help

Undocumented commands:
=====
EOF add subtract

calc >>> add
0
calc >>> add 4 5 6
15.0
calc >>> subtract
0
calc >>> subtract 10 2
8.0
calc >>> subtract 10 2 20
-12.0
calc >>> bye, bye

```

In this case we are using the prompt and intro class variables to

define what the default prompt looks like and a welcome message when the command interpreter is invoked.

In the `add` and `subtract` commands we are using the `strip` and `split` methods to parse all arguments. If you want to get fancy, you can use Python modules like `getopts` or `argparse` for this, but this is not necessary in this simple example.

### 17.6.15.3 Help Messages

Notice that all commands presently show up as undocumented. To remedy this, we can define `help_` methods for each command:

```
from __future__ import print_function, division
import cmd

class Calculator(cmd.Cmd):
 prompt = 'calc >>> '
 intro = 'Simple calculator that can do addition, subtraction, multiplication and division'

 def do_add(self, line):
 args = line.split()
 total = 0
 for arg in args:
 total += float(arg.strip())
 print(total)

 def help_add(self):
 print('\n'.join([
 'add [number,]',
 'Add the arguments together and display the total.'
]))

 def do_subtract(self, line):
 args = line.split()
 total = 0
 if len(args) > 0:
 total = float(args[0])
 for arg in args[1:]:
 total -= float(arg.strip())
 print(total)

 def help_subtract(self):
 print('\n'.join([
 'subtract [number,]',
 'Subtract all following arguments from the first argument.'
]))

 def do_EOF(self, line):
```

```
 print('bye, bye')
 return True

if __name__ == '__main__':
 Calculator().cmdloop()
```

Now, we can obtain help for the add and subtract commands:

```
$ python calculator.py
Simple calculator that can do addition, subtraction, multiplication and division.
calc >>> help

Documented commands (type help <topic>):
=====
add help subtract

Undocumented commands:
=====
EOF

calc >>> help add
add [number,]
Add the arguments together and display the total.
calc >>> help subtract
subtract [number,]
Subtract all following arguments from the first argument.
calc >>> bye, bye
```

#### 17.6.15.4 Useful Links

- [cms Python 2 Docs](#)
- [cmd Python 3 Docs](#)
- [Python Module of the Week: cmd – Create line-oriented command processors](#)
- [Python Module of the Week: cmd – Create line-oriented command processors](#)

#### 17.6.16 OPENCV



---

#### 🎓 Learning Objectives

- Provide some simple calculations so we can test cloud services.
  - Show case some elementary OpenCV functions
  - Show an environmental image analysis application using Secchi disks
- 

OpenCV (Open Source Computer Vision Library) is a library of thousands of algorithms for various applications in computer vision and machine learning. It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. In this section, we will explain basic features of this library, including the implementation of a simple example.

### **17.6.16.1 Overview**

OpenCV has countless functions for image and videos processing. The pipeline starts with reading the images, low-level operations on pixel values, preprocessing e.g. denoising, and then multiple steps of higher-level operations which vary depending on the application. OpenCV covers the whole pipeline, especially providing a large set of library functions for high-level operations. A simpler library for image processing in Python is Scipy's multi-dimensional image processing package (scipy.ndimage).

### **17.6.16.2 Installation**

OpenCV for Python can be installed on Linux in multiple ways, namely PyPI(Python Package Index), Linux package manager (apt-get for Ubuntu), Conda package manager, and also building from source. You are recommended to use PyPI. Here's the command that you need to run:

```
$ pip install opencv-python
```

This was tested on Ubuntu 16.04 with a fresh Python 3.6 virtual environment. In order to test, import the module in Python command

line:

```
import cv2
```

If it does not raise an error, it is installed correctly. Otherwise, try to solve the error.

For installation on Windows, see:

- [https://docs.opencv.org/3.0-beta/doc/doc/py\\_tutorials/py\\_setup/py\\_setup\\_in\\_windows/py\\_setup\\_opencv-python-in-windows](https://docs.opencv.org/3.0-beta/doc/doc/py_tutorials/py_setup/py_setup_in_windows/py_setup_opencv-python-in-windows)

Note that building from source can take a long time and may not be feasible for deploying to limited platforms such as Raspberry Pi.

### 17.6.16.3 A Simple Example

In this example, an image is loaded. A simple processing is performed, and the result is written to a new image.

#### 17.6.16.3.1 Loading an image

```
%matplotlib inline
import cv2

img = cv2.imread('images/opencv/4.2.01.tiff')
```

The image was downloaded from USC standard database:

<http://sipi.usc.edu/database/database.php?volume=misc&image=9>

#### 17.6.16.3.2 Displaying the image

The image is saved in a numpy array. Each pixel is represented with 3 values (R,G,B). This provides you with access to manipulate the image at the level of single pixels. You can display the image using imshow function as well as Matplotlib's imshow function.

You can display the image using imshow function:

```
cv2.imshow('Original',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

or you can use Matplotlib. If you have not installed Matplotlib before, install it using:

```
$ pip install matplotlib
```

Now you can use:

```
import matplotlib.pyplot as plt
plt.imshow(img)
```

which results in [Figure 146](#)

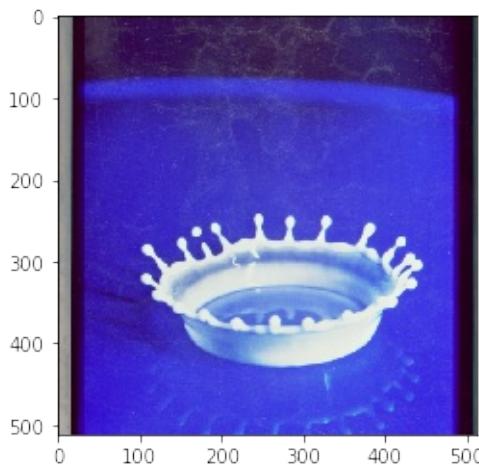


Figure 146: Image display

#### 17.6.16.3.3 Scaling and Rotation

Scaling (resizing) the image relative to different axis

```
res = cv2.resize(img,
 None,
 fx=1.2,
 fy=0.7,
 interpolation=cv2.INTER_CUBIC)
plt.imshow(res)
```

which results in [Figure 147](#)

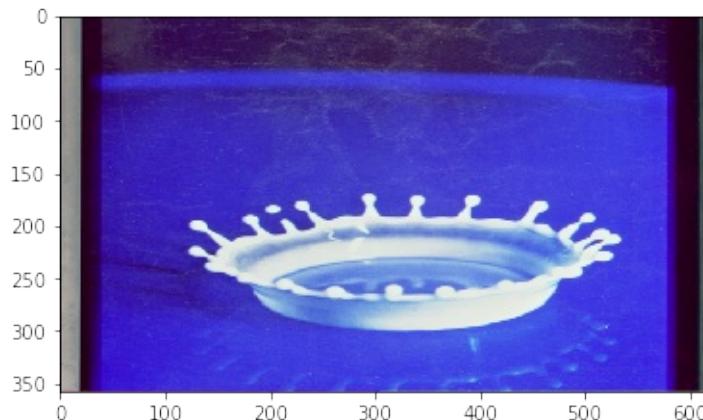


Figure 147: Scaling and rotation

Rotation of the image for an angle of t

```
rows,cols,_ = img.shape
t = 45
M = cv2.getRotationMatrix2D((cols/2,rows/2),t,1)
dst = cv2.warpAffine(img,M,(cols,rows))

plt.imshow(dst)
```

which results in [Figure 148](#)

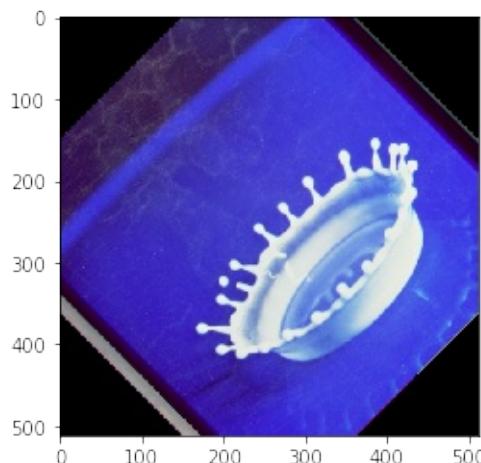


Figure 148: image

#### 17.6.16.3.4 Gray-scaling

```
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img2, cmap='gray')
```

which results in +[Figure 149](#)

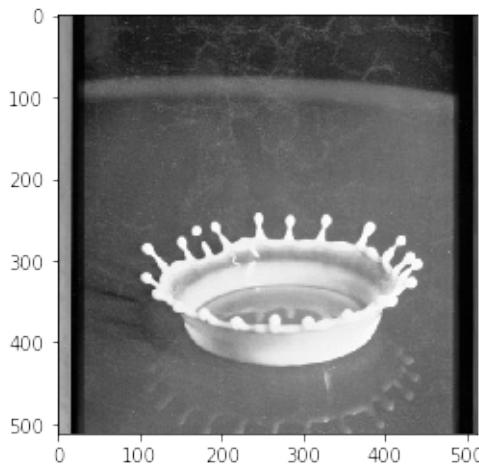


Figure 149: Gray scaling

#### 17.6.16.3.5 Image Thresholding

```
ret,thresh = cv2.threshold(img2,127,255, cv2.THRESH_BINARY)
plt.subplot(1,2,1), plt.imshow(img2, cmap='gray')
plt.subplot(1,2,2), plt.imshow(thresh, cmap='gray')
```

which results in [Figure 150](#)

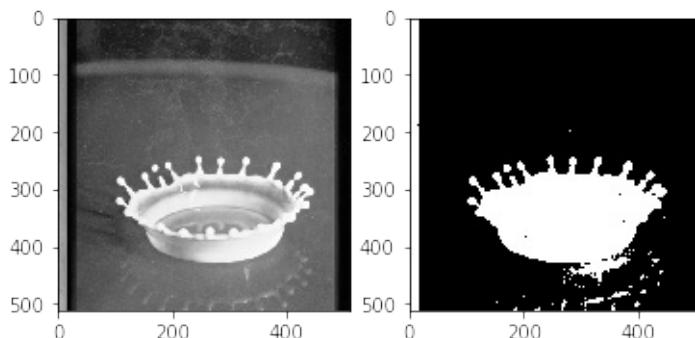


Figure 150: Image Thresholding

#### 17.6.16.3.6 Edge Detection

Edge detection using Canny edge detection algorithm

```
edges = cv2.Canny(img2,100,200)

plt.subplot(121),plt.imshow(img2,cmap = 'gray')
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
```

which results in [Figure 151](#)

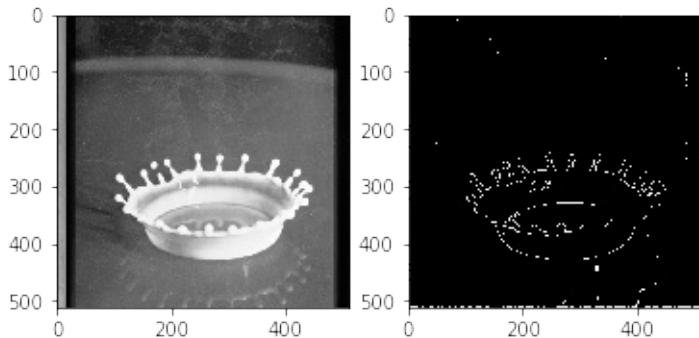


Figure 151: Edge detection

#### 17.6.16.4 Additional Features

OpenCV has implementations of many machine learning techniques such as KMeans and Support Vector Machines, that can be put into use with only a few lines of code. It also has functions especially for video analysis, feature detection, object recognition and many more. You can find out more about them in their website

[OpenCV](<https://docs.opencv.org/3.0-beta/index.html>) was initially developed for C++ and still has a focus on that language, but it is still one of the most valuable image processing libraries in Python.

#### 17.6.17 SECCHI DISK



We are developing an autonomous robot boat that you can be part of developing within this class. The robot bot is actually measuring turbidity or water clarity. Traditionally this has been done with a Secchi disk. The use of the Secchi disk is as follows:

1. Lower the Secchi disk into the water.
2. Measure the point when you can no longer see it
3. Record the depth at various levels and plot in a geographical 3D map

One of the things we can do is take a video of the measurement instead of a human recording them. Then we can analyse the video automatically to see how deep a disk was lowered. This is a classical

image analysis program. You are encouraged to identify algorithms that can identify the depth. The most simplest seems to be to do a histogram at a variety of depth steps, and measure when the histogram no longer changes significantly. The depth at that image will be the measurement we look for.

Thus if we analyse the images we need to look at the image and identify the numbers on the measuring tape, as well as the visibility of the disk.

To show case how such a disk looks like we refer to the image showcasing different Secchi disks. For our purpose the black-white contrast Secchi disk works well. See [Figure 152](#)

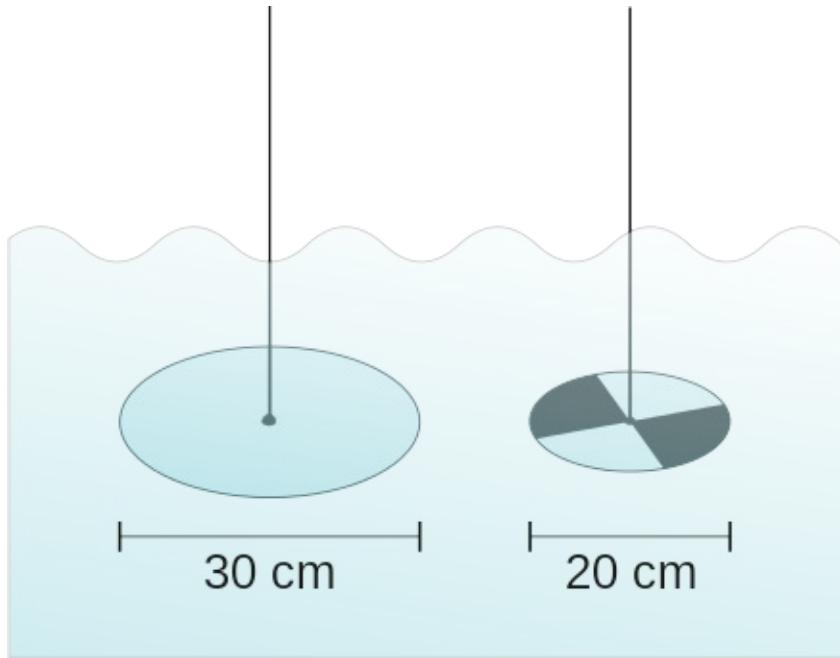


Figure 152: Secchi disk types. A marine style on the left and the freshwater version on the right wikipedia.

More information about Secchi Disk can be found at:

- [https://en.wikipedia.org/wiki/Secchi/\\_disk](https://en.wikipedia.org/wiki/Secchi_disk)

We have included next a couple of examples while using some obviously useful OpenCV methods. Surprisingly, the use of the edge detection that comes in mind first to identify if we still can see the

disk, seems to complicated to use for analysis. We at this time believe the histogram will be sufficient.

Please inspect our examples.

### 17.6.17.1 Setup for OSX

First let's setup the OpenCV environment for OSX. Naturally you will have to update the versions based on your versions of python. When we tried the install of OpenCV on MacOS, the setup was slightly more complex than other packages. This may have changed by now and if you have improved instructions, please let us know. However we do not want to install it via Anaconda out of the obvious reason that anaconda installs too many other things.

```
import os, sys
from os.path import expanduser
os.path
home = expanduser("~")
sys.path.append('/usr/local/Cellar/opencv/3.3.1_1/lib/python3.6/site-packages/')
sys.path.append(home + '/.pyenv/versions/OPENCV/lib/python3.6/site-packages/')
import cv2
cv2.__version__
! pip install numpy > tmp.log
! pip install matplotlib >> tmp.log
%matplotlib inline
```

### 17.6.17.2 Step 1: Record the video

Record the video on the robot

We have actually done this for you and will provide you with images and videos if you are interested in analyzing them. See [Figure 153](#)

### 17.6.17.3 Step 2: Analyse the images from the Video

For now we just selected 4 images from the video

```
import cv2
import matplotlib.pyplot as plt

img1 = cv2.imread('secchi/secchi1.png')
img2 = cv2.imread('secchi/secchi2.png')
```

```

img3 = cv2.imread('secchi/secchi3.png')
img4 = cv2.imread('secchi/secchi4.png')

figures = []
fig = plt.figure(figsize=(18, 16))
for i in range(1,13):
 figures.append(fig.add_subplot(4,3,i))
count = 0
for img in [img1,img2,img3,img4]:
 figures[count].imshow(img)

color = ('b','g','r')
for i,col in enumerate(color):
 histr = cv2.calcHist([img],[i],None,[256],[0,256])
 figures[count+1].plot(histr,color = col)

figures[count+2].hist(img.ravel(),256,[0,256])

count += 3

print("Legend")
print("First column = image of Secchi disk")
print("Second column = histogram of colors in image")
print("Third column = histogram of all values")

plt.show()

```

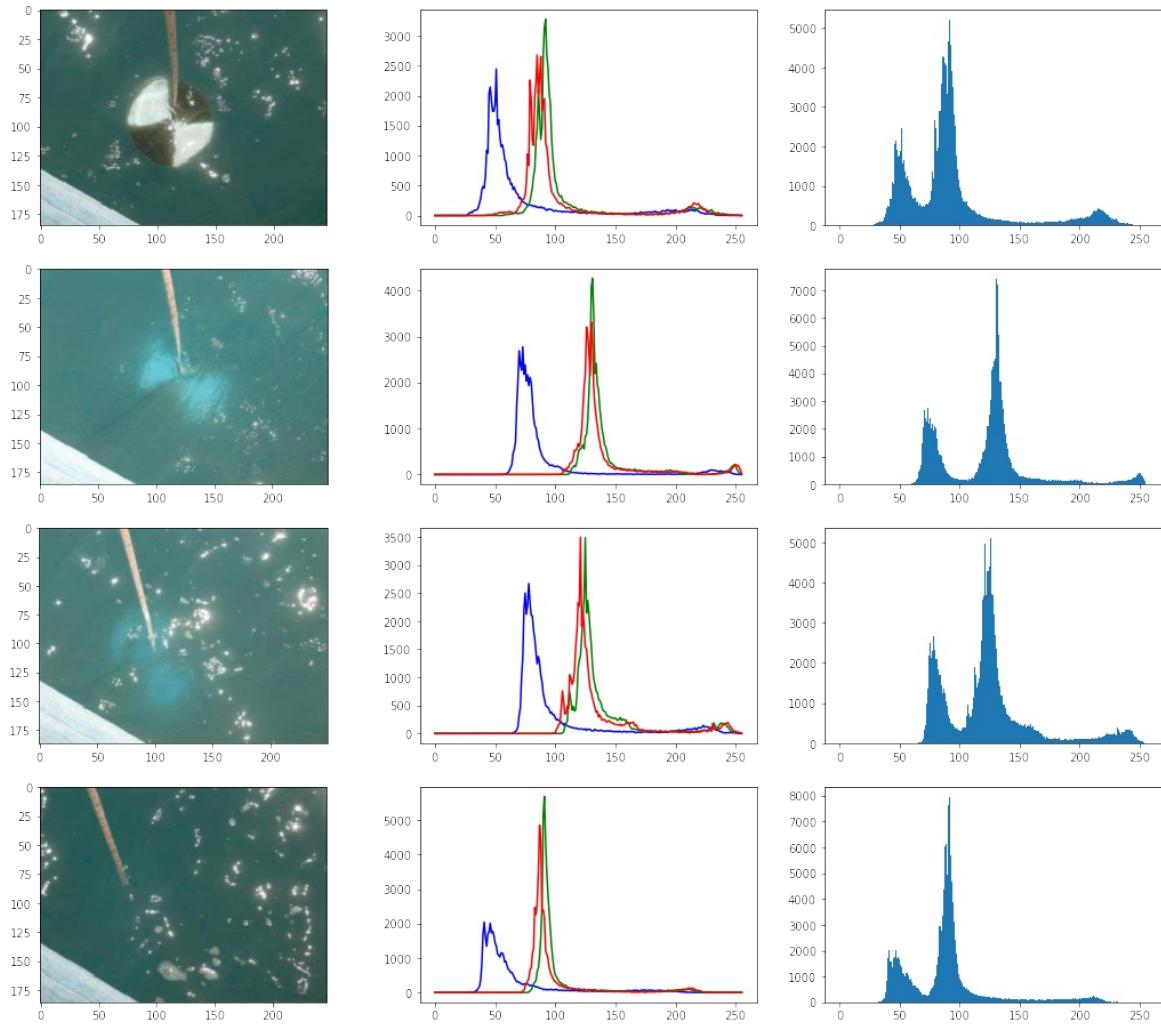


Figure 153: Histogram

#### 17.6.17.3.1 Image Thresholding

See [Figure 154](#), [Figure 155](#), [Figure 156](#), [Figure 157](#)

```
def threshold(img):
 ret,thresh = cv2.threshold(img,150,255,cv2.THRESH_BINARY)
 plt.subplot(1,2,1), plt.imshow(img, cmap='gray')
 plt.subplot(1,2,2), plt.imshow(thresh, cmap='gray')

threshold(img1)
```

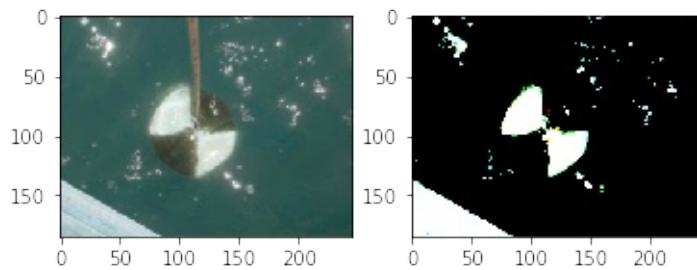


Figure 154: Threshold 1

```
threshold(img2)
```

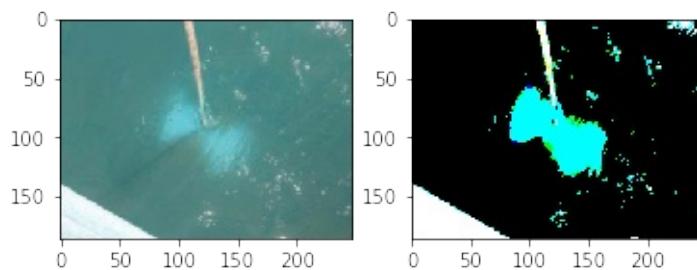


Figure 155: Threshold 2

```
threshold(img3)
```

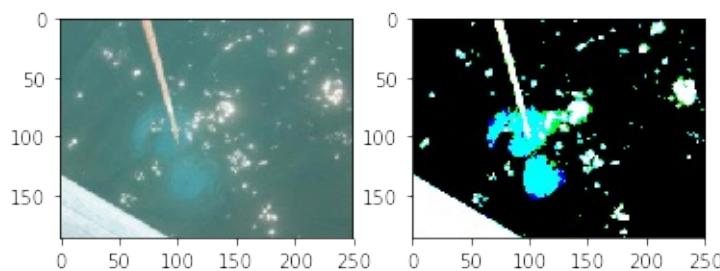


Figure 156: Threshold 3

```
threshold(img4)
```

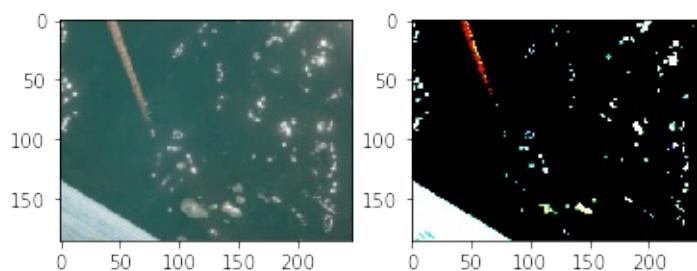


Figure 157: Threshold 4

### 17.6.17.3.2 Edge Detection

See [Figure 158](#), [Figure 159](#), [Figure 160](#), [Figure 161](#), [Figure 162](#). Edge detection using Canny edge detection algorithm

```
def find_edge(img):
 edges = cv2.Canny(img, 50, 200)
 plt.subplot(121),plt.imshow(img,cmap = 'gray')
 plt.subplot(122),plt.imshow(edges,cmap = 'gray')

find_edge(img1)
```

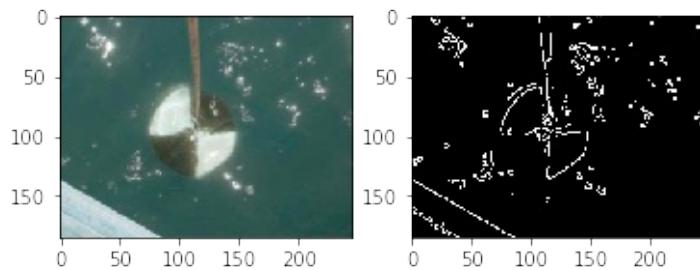


Figure 158: Edge Detection 1

```
find_edge(img2)
```

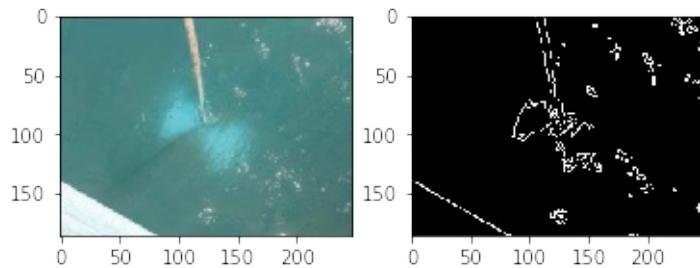


Figure 159: Edge Detection 2

```
find_edge(img3)
```

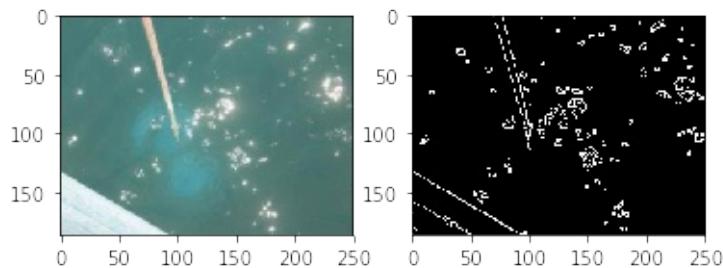


Figure 160: Edge Detection 3

```
find_edge(img4)
```

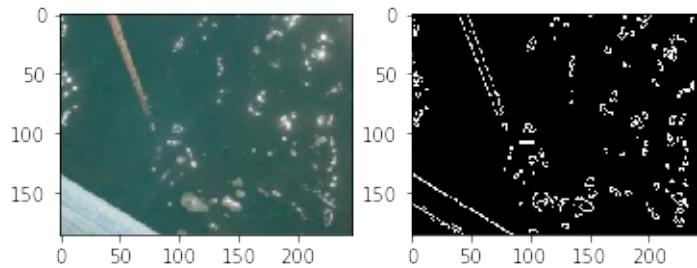


Figure 161: Edge Detection 4

#### 17.6.17.3.3 Black and white

```
bw1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
plt.imshow(bw1, cmap='gray')
```

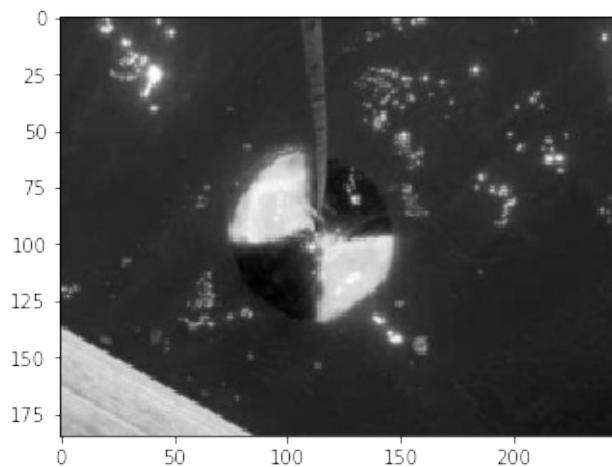


Figure 162: Back White conversion

### 17.6.18 DATA LIBRARIES



#### 17.6.18.1 DATA FORMATS



##### 17.6.18.1.1 YAML

The term YAML stand for “YAML Ainot Markup Language”. According to the Web Page at

- <http://yaml.org/>

“YAML is a human friendly data serialization standard for all programming languages.” There are multiple versions of YAML

existing and one needs to take care of that your software supports the right version. The current version is YAML 1.2.

YAML is often used for configuration and in many cases can also be used as XML replacement. Important is that YAML in contrast to XML removes the tags while replacing them with indentation. This has naturally the advantage that it is more easily to read, however, the format is strict and needs to adhere to proper indentation. Thus it is important that you check your YAML files for correctness, either by writing for example a python program that reads your yaml file, or an online YAML checker such as provided at

- <http://www.yamllint.com/>

An example on how to use yaml in python is provided in our next example. Please note that YAML is a superset of JSON. Originally YAML was designed as a markup language. However as it is not document oriented but data oriented it has been recast and it does no longer classify itself as markup language.

```
import os
import sys
import yaml

try:
 yamlFilename = os.sys.argv[1]
 yamlFile = open(yamlFilename, "r")
except:
 print("filename does not exist")
 sys.exit()
try:
 yaml.load(yamlFile.read())
except:
 print("YAML file is not valid.")
```

## Resources:

- <http://yaml.org/>
- <https://en.wikipedia.org/wiki/YAML>
- <http://www.yamllint.com/>

### 17.6.18.1.2 JSON

The term JSON stand for JavaScript Object Notation. It is targeted as an open-standard file format that emphasizes on integration of human-readable text to transmit data objects. The data objects contain attribute value pairs. Although it originates from JavaScript, the format itself is language independent. It uses brackets to allow organization of the data. Please note that YAML is a superset of JSON and not all YAML documents can be converted to JSON. Furthermore JSON does not support comments. For these reasons we often prefer to us YAML instead of JSON. However JSON data can easily be translated to YAML as well as XML.

Resources:

- <https://en.wikipedia.org/wiki/JSON>
- <https://www.json.org/>

#### 17.6.18.1.3 XML

XML stands for Extensible Markup Language. XML allows to define documents with the help of a set of rules in order to make it machine readable. The emphasize here is on machine readable as document in XML can become quickly complex and difficult to understand for humans. XML is used for documents as well as data structures.

A tutorial about XML is available at

- <https://www.w3schools.com/xml/default.asp>

Resources:

- <https://en.wikipedia.org/wiki/XML>

#### 17.6.18.2 MongoDB IN PYTHON



---

#### ❖ Learning Objectives

- Introduction to basic MongoDB knowledge

- Use of MOngoDB via PyMongo
  - Use of MongoEngine MongoEngine and Object-Document mapper,
  - Use of Flask-Mongo
- 

In today's era, NoSQL databases have developed an enormous potential to process the unstructured data efficiently. Modern information is complex, extensive, and may not have pre-existing relationships. With the advent of the advanced search engines, machine learning, and Artificial Intelligence, technology expectations to process, store, and analyze such data have grown tremendously [48]. The NoSQL database engines such as MongoDB, Redis, and Cassandra have successfully overcome the traditional relational database challenges such as scalability, performance, unstructured data growth, agile sprint cycles, and growing needs of processing data in real-time with minimal hardware processing power [49]. The NoSQL databases are a new generation of engines that do not necessarily require SQL language and are sometimes also called Not Only SQL databases. However, most of them support various third-party open connectivity drivers that can map NoSQL queries to SQL's. It would be safe to say that although NoSQL databases are still far from replacing the relational databases, they are adding an immense value when used in hybrid IT environments in conjunction with relational databases, based on the application specific needs [49]. In this paper, we will be covering the MongoDB technology, its driver PyMongo, its object-document mapper MongoEngine, and the Flask-PyMongo micro-web framework that make MongoDB more attractive and user-friendly.

#### **17.6.18.2.1 MongoDB**

Today MongoDB is one of leading NoSQL database which is fully capable of handling dynamic changes, processing large volumes of complex and unstructured data, easily using object-oriented programming features; as well as distributed system challenges [50]. At its core, MongoDB is an open source, cross-platform, document

database mainly written in C++ language.

#### 17.6.18.2.1.1 Installation

MongoDB can be installed on various Unix Platforms, including Linux, Ubuntu, Amazon Linux, etc [51]. This section focuses on installing MongoDB on Ubuntu 18.04 Bionic Beaver used as a standard OS for a virtual machine used as a part of Big Data Application Class during the 2018 Fall semester.

##### 17.6.18.2.1.1.1 Installation procedure

Before installing, it is recommended to configure the non-root user and provide the administrative privileges to it, in order to be able to perform general MongoDB admin tasks. This can be accomplished by login as the root user in the following manner [52].

```
$ adduser mongoadmin
$ usermod -aG sudo sammy
```

When logged in as a regular user, one can perform actions with superuser privileges by typing sudo before each command [52].

Once the user set up is completed, one can login as a regular user (mongoadmin) and use the following instructions to install MongoDB.

To update the Ubuntu packages to the most recent versions, use below command:

```
$ sudo apt update
```

To install the MongoDB package:

```
$ sudo apt install -y mongodb
```

To check the service and database status:

```
$ sudo systemctl status mongodb
```

Verifying the status of a successful MongoDB installation can be confirmed with an output similar to this:

```
$ mongodb.service - An object/document-oriented database
 Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: enabled)
 Active: **active** (running) since Sat 2018-11-15 07:48:04 UTC; 2min 17s ago
 Docs: man:mongod(1)
 Main PID: 2312 (mongod)
 Tasks: 23 (limit: 1153)
 CGroup: /system.slice/mongodb.service
 └─2312 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf
```

To verify the configuration, more specifically the installed version, server, and port, use the following command:

```
$ mongo --eval 'db.runCommand({ connectionStatus: 1 })'
```

Similarly, to restart MongoDB, use the following:

```
$ sudo systemctl restart mongodb
```

To allow access to MongoDB from an outside hosted server one can use the following command which opens the fire-wall connections [51].

```
$ sudo ufw allow from your_other_server_ip/32 to any port 27017
```

Status can be verified by using:

```
$ sudo ufw status
```

Other MongoDB configurations can be edited through the /etc/mongodb.conf files such as port and hostnames, file paths.

```
$ sudo nano /etc/mongodb.conf
```

Also, to complete this step, a server's IP address must be added to the bindIP value [51].

```
$ logappend=true

bind_ip = 127.0.0.1,your_server_ip
port = 27017
```

MongoDB is now listening for a remote connection that can be accessed by anyone with appropriate credentials [51].

#### 17.6.18.2.1.2 Collections and Documents

Each database within Mongo environment contains collections which in turn contain documents. Collections and documents are analogous to tables and rows respectively to the relational databases. The document structure is in a key-value form which allows storing of complex data types composed out of field and value pairs. Documents are objects which correspond to native data types in many programming languages, hence a well defined, embedded document can help reduce expensive joins and improve query performance. The `_id` field helps to identify each document uniquely [49].

MongoDB offers flexibility to write records that are not restricted by column types. The data storage approach is flexible as it allows one to store data as it grows and to fulfill varying needs of applications and/or users. It supports JSON like binary points known as BSON where data can be stored without specifying the type of data. Moreover, it can be distributed to multiple machines at high speed. It includes a sharding feature that partitions and spreads the data out across various servers. This makes MongoDB an excellent choice for cloud data processing. Its utilities can load high volumes of data at high speed which ultimately provides greater flexibility and availability in a cloud-based environment [48].

The dynamic schema structure within MongoDB allows easy testing of the small sprints in the Agile project management life cycles and research projects that require frequent changes to the data structure with minimal downtime. Contrary to this flexible process, modifying the data structure of relational databases can be a very tedious process [48].

#### 17.6.18.2.1.2.1 Collection example:

The following collection example for a person named Corey includes additional information such as age, status, and group [53].

```
{
 name: "Corey"
 age: "21"
 status: "Open"
 group: ["AI" , "Machine Learning"]
```

```
}
```

### 17.6.18.2.1.2.2 Document structure:

```
{
 field1: value1,
 field2: value2,
 field3: value3,
 ...
 fieldN: valueN
}
```

### 17.6.18.2.1.2.3 Collection Operations

If collection does not exists, MongoDB database will create a collection by default.

```
> db.myNewCollection1.insertOne({ x: 1 })
> db.myNewCollection2.createIndex({ y: 1 })
```

### 17.6.18.2.1.3 MongoDB Querying

The data retrieval patterns, the frequency of data manipulation statements such as insert, updates, and deletes may demand for the use of indexes or incorporating the sharding feature to improve query performance and efficiency of MongoDB environment [49]. One of the significant difference between relational databases and NoSQL databases are joins. In the relational database, one can combine results from two or more tables using a common column, often called as key. The native table contains the primary key column while the referenced table contains a foreign key. This mechanism allows one to make changes in a single row instead of changing all rows in the referenced table. This action is referred to as normalization. MongoDB is a document database and mainly contains denormalized data which means the data is repeated instead of indexed over a specific key. If the same data is required in more than one table, it needs to be repeated. This constraint has been eliminated in MongoDB's new version 3.2. The new release introduced a \$lookup feature which more likely works as a left-outer-join. Lookups are restricted to aggregated functions which means that data usually need some type of filtering and grouping operations to be conducted beforehand. For this reason, joins in MongoDB

require more complicated querying compared to the traditional relational database joins. Although at this time, lookups are still very far from replacing joins, this is a prominent feature that can resolve some of the relational data challenges for MongoDB [54]. MongoDB queries support regular expressions as well as range asks for specific fields that eliminate the need of returning entire documents [49]. MongoDB collections do not enforce document structure like SQL databases which is a compelling feature. However, it is essential to keep in mind the needs of the applications[48].

#### 17.6.18.2.1.3.1 Mongo Queries examples:

The queries can be executed from Mongo shell as well as through scripts.

To query the data from a MongoDB collection, one would use MongoDB's find() method.

```
> db.COLLECTION_NAME.find()
```

The output can be formatted by using the pretty() command.

```
> db.mycol.find().pretty()
```

The MongoDB insert statements can be performed in the following manner:

```
> db.COLLECTION_NAME.insert(document)
```

"The \$lookup command performs a left-outer-join to an unsharded collection in the same database to filter in documents from the joined collection for processing" [55].

```
$ {
 $lookup:
 {
 from: <collection to join>,
 localField: <field from the input documents>,
 foreignField: <field from the documents of the "from" collection>,
 as: <output array field>
 }
}
```

This operation is equivalent to the following SQL operation:

```
$ SELECT *, <output array field>
 FROM collection
 WHERE <output array field> IN (SELECT *
 FROM <collection to join>
 WHERE <foreignField> = <collection.localField>);
```

To perform a Like Match (Regex), one would use the following command:

```
> db.products.find({ sku: { $regex: /789$/ } })
```

#### 17.6.18.2.1.4 MongoDB Basic Functions

When it comes to the technical elements of MongoDB, it possesses a rich interface for importing and storage of external data in various formats. By using the Mongo Import/Export tool, one can easily transfer contents from JSON, CSV, or TSV files into a database. MongoDB supports CRUD (create, read, update, delete) operations efficiently and has detailed documentation available on the product website. It can also query the geospatial data, and it is capable of storing geospatial data in GeoJSON objects. The aggregation operation of the MongoDB process data records and returns computed results. MongoDB aggregation framework is modeled on the concept of data pipelines [56].

##### 17.6.18.2.1.4.1 Import/Export functions examples:

To import JSON documents, one would use the following command:

```
$ mongoimport --db users --collection contacts --file contacts.json
```

The CSV import uses the input file name to import a collection, hence, the collection name is optional [56].

```
$ mongoimport --db users --type csv --headerline --file /opt/backups/contacts.csv
```

“Mongoexport is a utility that produces a JSON or CSV export of data stored in a MongoDB instance” [56].

```
$ mongoexport --db test --collection traffic --out traffic.json
```

#### **17.6.18.2.1.5 Security Features**

Data security is a crucial aspect of the enterprise infrastructure management and is the reason why MongoDB provides various security features such as role based access control, numerous authentication options, and encryption. It supports mechanisms such as SCRAM, LDAP, and Kerberos authentication. The administrator can create role/collection-based access control; also roles can be predefined or custom. MongoDB can audit activities such as DDL, CRUD statements, authentication and authorization operations [57].

##### **17.6.18.2.1.5.1 Collection based access control example:**

A user defined role can contain the following privileges [57].

```
$ privileges: [
 { resource: { db: "products", collection: "inventory" }, actions: ["find", "update"] },
 { resource: { db: "products", collection: "orders" }, actions: ["find"] }
]
```

#### **17.6.18.2.1.6 MongoDB Cloud Service**

In regards to the cloud technologies, MongoDB also offers fully automated cloud service called Atlas with competitive pricing options. Mongo Atlas Cloud interface offers interactive GUI for managing cloud resources and deploying applications quickly. The service is equipped with geographically distributed instances to ensure no single point failure. Also, a well-rounded performance monitoring interface allows users to promptly detect anomalies and generate index suggestions to optimize the performance and reliability of the database. Global technology leaders such as Google, Facebook, eBay, and Nokia are leveraging MongoDB and Atlas cloud services making MongoDB one of the most popular choices among the NoSQL databases [58].

#### **17.6.18.2.2 PyMongo**

PyMongo is the official Python driver or distribution that allows work with a NoSQL type database called MongoDB [59]. The first version of

the driver was developed in 2009 [60], only two years after the development of MongoDB was started. This driver allows developers to combine both Python's versatility and MongoDB's flexible schema nature into successful applications. Currently, this driver supports MongoDB versions 2.6, 3.0, 3.2, 3.4, 3.6, and 4.0 [61]. MongoDB and Python represent a compatible fit considering that BSON (binary JSON) used in this NoSQL database is very similar to Python dictionaries, which makes the collaboration between the two even more appealing [62]. For this reason, dictionaries are the recommended tools to be used in PyMongo when representing documents [63].

#### 17.6.18.2.2.1 Installation

Prior to being able to exploit the benefits of Python and MongoDB simultaneously, the PyMongo distribution must be installed using pip. To install it on all platforms, the following command should be used [64]:

```
$ python -m pip install pymongo
```

Specific versions of PyMongo can be installed with command lines such as in our example where the 3.5.1 version is installed [64].

```
$ python -m pip install pymongo==3.5.1
```

A single line of code can be used to upgrade the driver as well [64].

```
$ python -m pip install --upgrade pymongo
```

Furthermore, the installation process can be completed with the help of the easy\_install tool, which requires users to use the following command [64].

```
$ python -m easy_install pymongo
```

To do an upgrade of the driver using this tool, the following command is recommended [64]:

```
$ python -m easy_install -U pymongo
```

There are many other ways of installing PyMongo directly from the source, however, they require for C extension dependencies to be installed prior to the driver installation step, as they are the ones that skim through the sources on GitHub and use the most up-to-date links to install the driver [64].

To check if the installation was completed accurately, the following command is used in the Python console [65].

```
import pymongo
```

If the command returns zero exceptions within the Python shell, one can consider for the PyMongo installation to have been completed successfully.

#### 17.6.18.2.2.2 Dependencies

The PyMongo driver has a few dependencies that should be taken into consideration prior to its usage. Currently, it supports CPython 2.7, 3.4+, PyPy, and PyPy 3.5+ interpreters [61]. An optional dependency that requires some additional components to be installed is the GSSAPI authentication [61]. For the Unix based machines, it requires pykerberos, while for the Windows machines WinKerberos is needed to fullfill this requirement [61]. The automatic installation of this dependency can be done simultaneously with the driver installation, in the following manner:

```
$ python -m pip install pymongo[gssapi]
```

Other third-party dependencies such as ipaddress, certifi, or wincertstore are necessary for connections with help of TLS/SSL and can also be simultaneously installed along with the driver installation [61].

#### 17.6.18.2.2.3 Running PyMongo with Mongo Deamon

Once PyMongo is installed, the Mongo deamon can be run with a very simple command in a new terminal window [65].

```
$ mongod
```

#### 17.6.18.2.2.4 Connecting to a database using MongoClient

In order to be able to establish a connection with a database, a MongoClient class needs to be imported, which subsequently allows the MongoClient object to communicate with the database [65].

```
from pymongo import MongoClient
client = MongoClient()
```

This command allows a connection with a default, local host through port 27017, however, depending on the programming requirements, one can also specify those by listing them in the client instance or use the same information via the Mongo URI format [65].

#### 17.6.18.2.2.5 Accessing Databases

Since MongoClient plays a server role, it can be used to access any desired databases in an easy way. To do that, one can use two different approaches. The first approach would be doing this via the attribute method where the name of the desired database is listed as an attribute, and the second approach, which would include a dictionary-style access [65]. For example, to access a database called cloudmesh\_community, one would use the following commands for the attribute and for the dictionary method, respectively.

```
db = client.cloudmesh_community
db = client['cloudmesh_community']
```

#### 17.6.18.2.2.6 Creating a Database

Creating a database is a straight forward process. First, one must create a MongoClient object and specify the connection (IP address) as well as the name of the database they are trying to create [66]. The example of this command is presented in the following section:

```
import pymongo
client = pymongo.MongoClient('mongodb://localhost:27017/')
db = client['cloudmesh']
```

#### 17.6.18.2.2.7 Inserting and Retrieving Documents (Querying)

Creating documents and storing data using PyMongo is equally easy as accessing and creating databases. In order to add new data, a collection must be specified first. In this example, a decision is made to use the cloudmesh group of documents.

```
cloudmesh = db.cloudmesh
```

Once this step is completed, data may be inserted using the `insert_one()` method, which means that only one document is being created. Of course, insertion of multiple documents at the same time is possible as well with use of the `insert_many()` method [65]. An example of this method is as follows:

```
course_info = {
 'course': 'Big Data Applications and Analytics',
 'instructor': ' Gregor von Laszewski',
 'chapter': 'technologies'
}
result = cloudmesh.insert_one(course_info)`
```

Another example of this method would be to create a collection. If we wanted to create a collection of students in the `cloudmesh_community`, we would do it in the following manner:

```
student = [{'name': 'John', 'st_id': 52642},
 {'name': 'Mercedes', 'st_id': 5717},
 {'name': 'Anna', 'st_id': 5654},
 {'name': 'Greg', 'st_id': 5423},
 {'name': 'Amaya', 'st_id': 3540},
 {'name': 'Cameron', 'st_id': 2343},
 {'name': 'Bozer', 'st_id': 4143},
 {'name': 'Cody', 'price': 2165}]

client = MongoClient('mongodb://localhost:27017/')

with client:
 db = client.cloudmesh
 db.students.insert_many(student)
```

Retrieving documents is equally simple as creating them. The `find_one()` method can be used to retrieve one document [65]. An implementation of this method is given in the following example.

```
gregors_course = cloudmesh.find_one({'instructor':'Gregor von Laszewski'})
```

Similarly, to retrieve multiple documents, one would use the find() method instead of the find\_one(). For example, to find all courses thought by professor von Laszewski, one would use the following command:

```
gregors_course = cloudmesh.find({'instructor': 'Gregor von Laszewski'})
```

One thing that users should be cognizant of when using the find() method is that it does not return results in an array format but as a cursor object, which is a combination of methods that work together to help with data querying [65]. In order to return individual documents, iteration over the result must be completed [65].

#### 17.6.18.2.2.8 Limiting Results

When it comes to working with large databases it is always useful to limit the number of query results. PyMongo supports this option with its limit() method [66]. This method takes in one parameter which specifies the number of documents to be returned [66]. For example, if we had a collection with a large number of cloud technologies as individual documents, one could modify the query results to return only the top 10 technologies. To do this, the following example could be utilized:

```
client = pymongo.MongoClient('mongodb://localhost:27017/')
db = client['cloudmesh']
col = db['technologies']
topten = col.find().limit(10)
```

#### 17.6.18.2.2.9 Updating Collection

Updating documents is very similar to inserting and retrieving the same. Depending on the number of documents to be updated, one would use the update\_one() or update\_many() method [66]. Two parameters need to be passed in the update\_one() method for it to successfully execute. The first argument is the query object that specifies the document to be changed, and the second argument is the object that specifies the new value in the document. An example of the update\_one() method in action is the following:

```
myquery = { 'course': 'Big Data Applications and Analytics' }
newvalues = { '$set': { 'course': 'Cloud Computing' } }
```

Updating all documents that fall under the same criteria can be done with the `update_many` method [66]. For example, to update all documents in which course title starts with letter B with a different instructor information, we would do the following:

```
client = pymongo.MongoClient('mongodb://localhost:27017/')
db = client['cloudmesh']
col = db['courses']
query = { 'course': { '$regex': '^B' } }
newvalues = { '$set': { 'instructor': 'Gregor von Laszewski' } }

edited = col.update_many(query, newvalues)
```

#### 17.6.18.2.2.10 Counting Documents

Counting documents can be done with one simple operation called `count_documents()` instead of using a full query [67]. For example, we can count the documents in the `cloudmesh_community` by using the following command:

```
cloudmesh = count_documents({})
```

To create a more specific count, one would use a command similar to this:

```
cloudmesh = count_documents({'author': 'von Laszewski'})
```

This technology supports some more advanced querying options as well. Those advanced queries allow one to add certain constraints and narrow down the results even more. For example, to get the courses thought by professor von Laszewski after a certain date, one would use the following command:

```
d = datetime.datetime(2017, 11, 12, 12)
for course in cloudmesh.find({'date': {'$lt': d}}).sort('author'):
 pprint.pprint(course)
```

#### 17.6.18.2.2.11 Indexing

Indexing is a very important part of querying. It can greatly improve query performance but also add functionality and aide in storing

documents [67].

"To create a unique index on a key that rejects documents whose value for that key already exists in the index" [67].

We need to firstly create the index in the following manner:

```
result = db.profiles.create_index([('user_id', pymongo.ASCENDING)],
unique=True)
sorted(list(db.profiles.index_information()))
```

This command acutally creates two different indexes. The first one is the \*\_id\* , created by MongoDB automatically, and the second one is the user\_id, created by the user.

The purpose of those indexes is to cleverly prevent future additions of invalid user\_ids into a collection.

#### 17.6.18.2.2.12 Sorting

Sorting on the server-side is also avaialable via MongoDB. The PyMongo sort() method is equivalent to the SQL order by statement and it can be performed as pymongoascending and pymongo.descending [68]. This method is much more efficient as it is being completed on the server-side, compared to the sorting completed on the client side. For example, to return all users with first name Gregor sorted in descending order by birthdate we would use a command such as this:

```
users = cloudmesh.users.find({'firstname':'Gregor'}).sort(('dateofbirth', pymongo.DESCENDING))
for user in users:
 print user.get('email')
```

#### 17.6.18.2.2.13 Aggregation

Aggregation operations are used to process given data and produce summarized results. Aggregation operations collect data from a number of documents and provide collective results by grouping data. PyMongo in its documentation offers a separate framework that

supports data aggregation. This aggregation framework can be used to

“provide projection capabilities to reshape the returned data” [69].

In the aggregation pipeline, documents pass through multiple pipeline stages which convert documents into result data. The basic pipeline stages include filters. Those filters act like document transformation by helping change the document output form. Other pipelines help group or sort documents with specific fields. By using native operations from MongoDB, the pipeline operators are efficient in aggregating results.

The addFields stage is used to add new fields into documents. It reshapes each document in stream, similarly to the project stage. The output document will contain existing fields from input documents and the newly added fields [70]. The following example shows how to add student details into a document.

```
db.cloudmesh_community.aggregate([
 {
 $addFields: {
 "document.StudentDetails": {
 $concat:['$document.student.FirstName', '$document.student.LastName']
 }
 }
 }
])
```

The bucket stage is used to categorize incoming documents into groups based on specified expressions. Those groups are called buckets [70]. The following example shows the bucket stage in action.

```
db.user.aggregate([
 { "$group": {
 "_id": {
 "city": "$city",
 "age": {
 "$let": {
 "vars": {
 "age": { "$subtract": [{ "$year": new Date() }, { "$year": "$birthDay" }] }
 }
 }
 }
 }
 }
])
```

```

 { "case": { "$lt": ["$$age", 40] }, "then": 30 },
 { "case": { "$lt": ["$$age", 50] }, "then": 40 },
 { "case": { "$lt": ["$$age", 200] }, "then": 50 }
] } } } } },
"count": { "$sum": 1 } }})

```

In the `bucketAuto` stage, the boundaries are automatically determined in an attempt to evenly distribute documents into a specified number of buckets. In the following operation, input documents are grouped into four buckets according to the values in the `price` field [70].

```

db.artwork.aggregate([
 {
 $bucketAuto: {
 groupBy: "$price",
 buckets: 4
 }
 }
])

```

The `collStats` stage returns statistics regarding a collection or view [70].

```

db.matrices.aggregate([{ $collStats: { latencyStats: { histograms: true } } }
])

```

The `count` stage passes a document to the next stage that contains the number documents that were input to the stage [70].

```

db.scores.aggregate([{
 $match: { score: { $gt: 80 } } },
 { $count: "passing_scores" }])

```

The `facet` stage helps process multiple aggregation pipelines in a single stage [70].

```

db.artwork.aggregate([{
 $facet: { "categorizedByTags": [{ $unwind: "$tags" },
 { $sortByCount: "$tags" }], "categorizedByPrice": [
 // Filter out documents without a price e.g., _id: 7
 { $match: { price: { $exists: 1 } } },
 { $bucket: { groupBy: "$price",
 boundaries: [0, 150, 200, 300, 400],
 default: "Other",
 output: { "count": { $sum: 1 },
 "titles": { $push: "$title" }
 } }],
 "categorizedByYears(Auto)": [
 { $bucketAuto: { groupBy: "$year", buckets: 4 }
] }] } })

```

The geoNear stage returns an ordered stream of documents based on the proximity to a geospatial point. The output documents include an additional distance field and can include a location identifier field [70].

```
db.places.aggregate([
 {
 $geoNear: {
 near: { type: "Point", coordinates: [-73.99279 , 40.719296] },
 distanceField: "dist.calculated",
 maxDistance: 2,
 query: { type: "public" },
 includeLocs: "dist.location",
 num: 5,
 spherical: true
 }
 }
])
```

The graphLookup stage performs a recursive search on a collection. To each output document, it adds a new array field that contains the traversal results of the recursive search for that document [70].

```
db.travelers.aggregate([
 {
 $graphLookup: {
 from: "airports",
 startWith: "$nearestAirport",
 connectFromField: "connects",
 connectToField: "airport",
 maxDepth: 2,
 depthField: "numConnections",
 as: "destinations"
 }
 }
])
```

The group stage consumes the document data per each distinct group. It has a RAM limit of 100 MB. If the stage exceeds this limit, the group produces an error [70].

```
db.sales.aggregate(
 [
 {
 $group : {
 _id : { month: { $month: "$date" }, day: { $dayOfMonth: "$date" },
 year: { $year: "$date" } },
 totalPrice: { $sum: { $multiply: ["$price", "$quantity"] } },
 averageQuantity: { $avg: "$quantity" },
 count: { $sum: 1 }
 }
 }
]
)
```

The indexStats stage returns statistics regarding the use of each index for a collection [70].

```
db.orders.aggregate([{ $indexStats: {} }])
```

The limit stage is used for controlling the number of documents passed to the next stage in the pipeline [70].

```
db.article.aggregate(
 { $limit : 5 }
)
```

The listLocalSessions stage gives the session information currently connected to mongos or mongod instance [70].

```
db.aggregate([{ $listLocalSessions: { allUsers: true } }])
```

The listSessions stage lists out all session that have been active long enough to propagate to the system.sessions collection [70].

```
use config
db.system.sessions.aggregate([{ $listSessions: { allUsers: true } }])
```

The lookup stage is useful for performing outer joins to other collections in the same database [70].

```
{
 $lookup:
 {
 from: <collection to join>,
 localField: <field from the input documents>,
 foreignField: <field from the documents of the "from" collection>,
 as: <output array field>
 }
}
```

The match stage is used to filter the document stream. Only matching documents pass to next stage [70].

```
db.articles.aggregate(
 [{ $match : { author : "dave" } }]
)
```

The project stage is used to reshape the documents by adding or deleting the fields.

```
db.books.aggregate([{ $project : { title : 1 , author : 1 } }])
```

The redact stage reshapes stream documents by restricting information using information stored in documents themselves [70].

```
db.accounts.aggregate(
[
 { $match: { status: "A" } },
 {
 $redact: {
 $cond: {
 if: { $eq: ["$level", 5] },
 then: "$$PRUNE",
 else: "$$DESCEND"
 } } }]);
```

The replaceRoot stage is used to replace a document with a specified embedded document [70].

```
db.produce.aggregate([
 {
 $replaceRoot: { newRoot: "$in_stock" }
 }
])
```

The sample stage is used to sample out data by randomly selecting number of documents from input [70].

```
db.users.aggregate(
 [{ $sample: { size: 3 } }]
)
```

The skip stage skips specified initial number of documents and passes remaining documents to the pipeline [70].

```
db.article.aggregate(
 { $skip : 5 }
)
```

The sort stage is useful while reordering document stream by a specified sort key [70].

```
db.users.aggregate(
 [
 { $sort : { age : -1, posts: 1 } }
]
)
```

The sortByCounts stage groups the incoming documents based on a specified expression value and counts documents in each distinct

group [70].

```
db.exhibits.aggregate(
[{ $unwind: "$tags" }, { $sortByCount: "$tags" }])
```

The unwind stage deconstructs an array field from the input documents to output a document for each element [70].

```
db.inventory.aggregate([{ $unwind: "$sizes" }])
db.inventory.aggregate([{ $unwind: { path: "$sizes" } }])
```

The out stage is used to write aggregation pipeline results into a collection. This stage should be the last stage of a pipeline [70].

```
db.books.aggregate([
 { $group : { _id : "$author", books: { $push: "$title" } } },
 { $out : "authors" }
])
```

Another option from the aggregation operations is the Map/Reduce framework, which essentially includes two different functions, map and reduce. The first one provides the key value pair for each tag in the array, while the latter one

“sums over all of the emitted values for a given key” [69].

The last step in the Map/Reduce process it to call the map\_reduce() function and iterate over the results [69]. The Map/Reduce operation provides result data in a collection or returns results in-line. One can perform subsequent operations with the same input collection if the output of the same is written to a collection [71]. An operation that produces results in a in-line form must provide results with in the BSON document size limit. The current limit for a BSON document is 16 MB. These types of operations are not supported by views [71]. The PyMongo’s API supports all features of the MongoDB’s Map/Reduce engine [72]. Moreover, Map/Reduce has the ability to get more detailed results by passing full\_response=True argument to the map\_reduce() function [72].

#### 17.6.18.2.2.14 Deleting Documents from a Collection

The deletion of documents with PyMongo is fairly straight forward. To do so, one would use the remove() method of the PyMongo Collection object [68]. Similarly to the reads and updates, specification of documents to be removed is a must. For example, removal of the entire document collection with a score of 1, would required one to use the following command:

```
cloudmesh.users.remove({"score":1, safe=True})
```

The safe parameter set to True ensures the operation was completed [68].

#### 17.6.18.2.2.15 Copying a Database

Copying databases within the same mongod instance or between different mongod servers is made possible with the command() method after connecting to the desired mongod instance [73]. For example, to copy the cloudmesh database and name the new database cloudmesh\_copy, one would use the command() method in the following manner:

```
client.admin.command('copydb',
 fromdb='cloudmesh',
 todb='cloudmesh_copy')
```

There are two ways to copy a database between servers. If a server is not password-protected, one would not need to pass in the credentials nor to authenticate to the admin database [73]. In that case, to copy a database one would use the following command:

```
client.admin.command('copydb',
 fromdb='cloudmesh',
 todb='cloudmesh_copy',
 fromhost='source.example.com')
```

On the other hand, if the server where we are copying the database to is protected, one would use this command instead:

```
client = MongoClient('target.example.com',
 username='administrator',
 password='pwd')
client.admin.command('copydb',
 fromdb='cloudmesh',
```

```
todb='cloudmesh_copy',
fromhost='source.example.com')
```

#### 17.6.18.2.2.16 PyMongo Strengths

One of PyMongo strengths is that allows document creation and querying natively

“through the use of existing language features such as nested dictionaries and lists” [68].

For moderately experienced Python developers, it is very easy to learn it and quickly feel comfortable with it.

“For these reasons, MongoDB and Python make a powerful combination for rapid, iterative development of horizontally scalable backend applications” [68].

According to [68], MongoDB is very applicable to modern applications, which makes PyMongo equally valuable [68].

#### 17.6.18.2.3 MongoEngine

“MongoEngine is an Object-Document Mapper, written in Python for working with MongoDB” [74].

It is actually a library that allows a more advanced communication with MongoDB compared to PyMongo. As MongoEngine is technically considered to be an object-document mapper(ODM), it can also be considered to be

“equivalent to a SQL-based object relational mapper(ORM)” [65].

The primary technique why one would use an ODM includes data conversion between computer systems that are not compatible with each other [75]. For the purpose of converting data to the appropriate form, a virtual object database must be created within the utilized programming language [75]. This library is also used to

define schemata for documents within MongoDB, which ultimately helps with minimizing coding errors as well defining methods on existing fields [76]. It is also very beneficial to the overall workflow as it tracks changes made to the documents and aids in the document saving process [77].

#### 17.6.18.2.3.1 Installation

The installation process for this technology is fairly simple as it is considered to be a library. To install it, one would use the following command [78]:

```
$ pip install mongoengine
```

A bleeding-edge version of MongoEngine can be installed directly from GitHub by first cloning the repository on the local machine, virtual machine, or cloud.

#### 17.6.18.2.3.2 Connecting to a database using MongoEngine

Once installed, MongoEngine needs to be connected to an instance of the mongod, similarly to PyMongo [79]. The connect() function must be used to successfully complete this step and the argument that must be used in this function is the name of the desired database [79]. Prior to using this function, the function name needs to be imported from the MongoEngine library.

```
from mongoengine import connect
connect('cloudmesh_community')
```

Similarly to the MongoClient, MongoEngine uses the local host and port 27017 by default, however, the connect() function also allows specifying other hosts and port arguments as well [79].

```
connect('cloudmesh_community', host='196.185.1.62', port=16758)
```

Other types of connections are also supported (i.e. URI) and they can be completed by providing the URI in the connect() function [79].

#### 17.6.18.2.3.3 Querying using MongoEngine

To query MongoDB using MongoEngine an objects attribute is used, which is, technically, a part of the document class [80]. This attribute is called the QuerySetManager which in return

"creates a new QuerySet object on access" [80].

To be able to access individual documents from a database, this object needs to be iterated over. For example, to return/print all students in the cloudmesh\_community object (database), the following command would be used.

```
for user in cloudmesh_community.objects:
 print cloudmesh_community.student
```

MongoEngine also has a capability of query filtering which means that a keyword can be used within the called QuerySet object to retrieve specific information [80]. Let's say one would like to iterate over cloudmesh\_community students that are natives of Indiana. To achieve this, one would use the following command:

```
indy_students = cloudmesh_community.objects(state='IN')
```

This library also allows the use of all operators except for the equality operator in its queries, and moreover, has the capability of handling string queries, geo queries, list querying, and querying of the raw PyMongo queries [80].

The string queries are useful in performing text operations in the conditional queries. A query to find a document exactly matching and with state ACTIVE can be performed in the following manner:

```
db.cloudmesh_community.find(State.exact("ACTIVE"))
```

The query to retrieve document data for names that start with a case sensitive AL can be written as:

```
db.cloudmesh_community.find(Name.startswith("AL"))
```

To perform an exact same query for the non-key-sensitive AL one would use the following command:

```
db.cloudmesh_community.find(Name.istartswith("AL"))
```

The MongoEngine allows data extraction of geographical locations by using Geo queries. The geo\_within operator checks if a geometry is within a polygon.

```
cloudmesh_community.objects(
 point_geo_within=[[{"type": "Polygon",
 "coordinates": [[[40, 5], [40, 6], [41, 6], [40, 5]]]}])
cloudmesh_community.objects(
 point_geo_within={"type": "Polygon",
 "coordinates": [[[40, 5], [40, 6], [41, 6], [40, 5]]]})
```

The list query looks up the documents where the specified fields matches exactly to the given value. To match all pages that have the word coding as an item in the tags list one would use the following query:

```
class Page(Document):
 tags = ListField(StringField())

Page.objects(tags='coding')
```

Overall, it would be safe to say that MongoEngine has good compatibility with Python. It provides different functions to utilize Python easily with MongoDB which makes this pair even more attractive to application developers.

#### 17.6.18.2.4 Flask-PyMongo

“Flask is a micro-web framework written in Python” [\[81\]](#).

It was developed after Django, and it is very pythonic in nature which implies that it is explicitly targeting the Python user community. It is lightweight as it does not require additional tools or libraries and hence is classified as a Micro-Web framework. It is often used with MongoDB using PyMongo connector, and it treats data within MongoDB as searchable Python dictionaries. The applications such as Pinterest, LinkedIn, and the community web page for Flask are using the Flask framework. Moreover, it supports various features such as the RESTful request dispatching, secure cookies, Google app engine compatibility, and integrated support for unit testing, etc [\[81\]](#). When it

comes to connecting to a database, the connection details for MongoDB can be passed as a variable or configured in PyMongo constructor with additional arguments such as username and password, if required. It is important that versions of both Flask and MongoDB are compatible with each other to avoid functionality breaks [82].

#### 17.6.18.2.4.1 Installation

Flask-PyMongo can be installed with an easy command such as this:

```
$ pip install Flask-PyMongo
```

PyMongo can be added in the following manner:

```
from flask import Flask
from flask_pymongo import PyMongo
app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://localhost:27017/cloudmesh_community"
mongo = PyMongo(app)
```

#### 17.6.18.2.4.2 Configuration

There are two ways to configure Flask-PyMongo. The first way would be to pass a MongoDB URI to the PyMongo constructor, while the second way would be to

“assign it to the MONGO\_URI Flask configuration variable” [82].

#### 17.6.18.2.4.3 Connection to multiple databases/servers

Multiple PyMongo instances can be used to connect to multiple databases or database servers. To achieve this, one would use a command similar to the following:

```
app = Flask(__name__)
mongo1 = PyMongo(app, uri="mongodb://localhost:27017/cloudmesh_community_one")
mongo2 = PyMongo(app, uri="mongodb://localhost:27017/cloudmesh_community_two")
mongo3 = PyMongo(app, uri=
 "mongodb://another.host:27017/cloudmesh_community_Three")
```

#### 17.6.18.2.4.4 Flask-PyMongo Methods

Flask-PyMongo provides helpers for some common tasks. One of them is the `Collection.find_one_or_404` method shown in the following example:

```
@app.route("/user/<username>")
def user_profile(username):
 user = mongo.db.cloudmesh_community.find_one_or_404({"_id": username})
 return render_template("user.html", user=user)
```

This method is very similar to the MongoDB's `find_one()` method, however, instead of returning `None` it causes a 404 Not Found HTTP status [82].

Similarly, the `PyMongo.send_file` and `PyMongo.save_file` methods work on the file-like objects and save them to GridFS using the given file name [82].

#### 17.6.18.2.4.5 Additional Libraries

Flask-MongoAlchemy and Flask-MongoEngine are the additional libraries that can be used to connect to a MongoDB database while using enhanced features with the Flask app. The Flask-MongoAlchemy is used as a proxy between Python and MongoDB to connect. It provides an option such as server or database based authentication to connect to MongoDB. While the default is set server based, to use a database-based authentication, the config value `MONGOALCHEMY_SERVER_AUTH` parameter must be set to `False` [83].

Flask-MongoEngine is the Flask extension that provides integration with the MongoEngine. It handles connection management for the apps. It can be installed through pip and set up very easily as well. The default configuration is set to the local host and port 27017. For the custom port and in cases where MongoDB is running on another server, the host and port must be explicitly specified in connect strings within the `MONGODB_SETTINGS` dictionary with `app.config`, along with the database username and password, in cases where a database authentication is enabled. The URI style connections are also supported and supply the URI as the host in the

MONGODB\_SETTINGS dictionary with app.config. There are various custom query sets that are available within Flask-Mongoengine that are attached to Mongoengine's default queryset [84].

#### 17.6.18.2.4.6 Classes and Wrappers

Attributes such as cx and db in the PyMongo objects are the ones that help provide access to the MongoDB server [82]. To achieve this, one must pass the Flask app to the constructor or call init\_app() [82].

"Flask-PyMongo wraps PyMongo's MongoClient, Database, and Collection classes, and overrides their attribute and item accessors" [82].

This type of wrapping allows Flask-PyMongo to add methods to Collection while at the same time allowing a MongoDB-style dotted expressions in the code [82].

```
type(mongo.cx)
type(mongo.db)
type(mongo.db.cloudmesh_community)
```

Flask-PyMongo creates connectivity between Python and Flask using a MongoDB database and supports

"extensions that can add application features as if they were implemented in Flask itself" [85],

hence, it can be used as an additional Flask functionality in Python code. The extensions are there for the purpose of supporting form validations, authentication technologies, object-relational mappers and framework related tools which ultimately adds a lot of strength to this micro-web framework [85]. One of the main reasons and benefits why it is frequently used with MongoDB is its capability of adding more control over databases and history [85].

### 17.6.18.3 MONGOENGINE



#### 17.6.18.3.1 Introduction

MongoEngine is a document mapper for working with mongodb with python. To be able to use mongo engine MongodD should be already installed and running.

#### 17.6.18.3.2 Install and connect

Mongoengine can be installed by running:

```
$ pip install mongo engine
```

This will install six, pymongo and mongoengine.

To connect to mongoldb use connect () function by specifying mongoldb instance name. You don't need to go to mongo shell but this can be done from unix shell or cmd line. In this case we are connecting to a database named student\_db.

```
from mongoengine import * connect ('student_db')
```

If mongodb is running on a port different from default port , port number and host need to be specified. If mongodb needs authentication username and password need to be specified.

#### 17.6.18.3.3 Basics

Mongodb does not enforce schemas. Comparing to RDBMS, Row in mongoldb is called a "document" and table can be compared to Collection. Defining a schema is helpful as it minimizes coding error's. To define a schema we create a class that inherits from document.

```
from mongoengine import *

class Student(Document):
 first_name = StringField(max_length=50)
 last_name = StringField(max_length=50)
```

○ TODO: Can you fix the code sections and look at the examples we provided.

Fields are not mandatory but if needed, set the required keyword argument to True. There are multiple values available for field types.

Each field can be customized by keyword argument. If each student is sending text messages to Universities central database , these can be stored using Mongodb. Each text can have different data types, some might have images or some might have url's. So we can create a class text and link it to student by using Reference field (similar to foreign key in RDBMS).

```
class Text(Document):
 title = StringField(max_length=120, required=True)
 author = ReferenceField(Student)
 meta = {'allow_inheritance': True}

class OnlyText(Text):
 content = StringField()

class ImagePost(Text):
 image_path = StringField()

class LinkPost(Text):
 link_url = StringField()
```

MongoDb supports adding tags to individual texts rather than storing them separately and then having them referenced. Similarly Comments can also be stored directly in a Text.

```
class Text(Document):
 title = StringField(max_length=120, required=True)
 author = ReferenceField(User)
 tags = ListField(StringField(max_length=30))
 comments = ListField(EmbeddedDocumentField(Comment))
```

For accessing data: if we need to get titles.

```
for text in OnlyText.objects:
 print(text.title)
```

Searching texts with tags.

```
for text in Text.objects(tags='mongodb'):
 print(text.title)
```

## 17.7 WORD COUNT WITH PARALLEL PYTHON



We will demonstrate Python's `multiprocessing` API for parallel computation by writing a program that counts how many times each word in a collection of documents appear.

## 17.7.1 Generating a Document Collection

Before we begin, let us write a script that will generate document collections by specifying the number of documents and the number of words per document. This will make benchmarking straightforward.

To keep it simple, the vocabulary of the document collection will consist of random numbers rather than the words of an actual language:

```
'''Usage: generate_nums.py [-h] NUM_LISTS INTS_PER_LIST MIN_INT MAX_INT DEST_DIR

Generate random lists of integers and save them
as 1.txt, 2.txt, etc.

Arguments:
 NUM_LISTS The number of lists to create.
 INTS_PER_LIST The number of integers in each list.
 MIN_NUM Each generated integer will be >= MIN_NUM.
 MAX_NUM Each generated integer will be <= MAX_NUM.
 DEST_DIR A directory where the generated numbers will be stored.

Options:
 -h --help
'''

from __future__ import print_function
import os, random, logging
from docopt import docopt

def generate_random_lists(num_lists,
 ints_per_list, min_int, max_int):
 return [[random.randint(min_int, max_int) \
 for i in range(ints_per_list)] for i in range(num_lists)]

if __name__ == '__main__':
 args = docopt(__doc__)
 num_lists, ints_per_list, min_int, max_int, dest_dir = [
 int(args['NUM_LISTS']),
 int(args['INTS_PER_LIST']),
 int(args['MIN_INT']),
 int(args['MAX_INT']),
 args['DEST_DIR']
]

 if not os.path.exists(dest_dir):
 os.makedirs(dest_dir)
```

```

lists = generate_random_lists(num_lists,
 ints_per_list,
 min_int,
 max_int)
curr_list = 1
for lst in lists:
 with open(os.path.join(dest_dir, '%d.txt' % curr_list), 'w') as f:
 f.write(os.linesep.join(map(str, lst)))
curr_list += 1
logging.debug('Numbers written.')

```

Notice that we are using the [docopt](#) module that you should be familiar with from the Section [Python DocOpts](#s-python-docopts) to make the script easy to run from the command line.

You can generate a document collection with this script as follows:

```
python generate_nums.py 1000 10000 0 100 docs-1000-10000
```

## 17.7.2 Serial Implementation

A first serial implementation of wordcount is straightforward:

```

'''Usage: wordcount.py [-h] DATA_DIR

Read a collection of .txt documents and count how many times each word
appears in the collection.

Arguments:
 DATA_DIR A directory with documents (.txt files).

Options:
 -h --help
 ...

from __future__ import division, print_function
import os, glob, logging
from docopt import docopt

logging.basicConfig(level=logging.DEBUG)

def wordcount(files):
 counts = {}
 for filepath in files:
 with open(filepath, 'r') as f:
 words = [word.strip() for word in f.read().split()]
 for word in words:
 if word not in counts:
 counts[word] = 0
 counts[word] += 1
 return counts

```

```
if __name__ == '__main__':
 args = docopt(__doc__)
 if not os.path.exists(args['DATA_DIR']):
 raise ValueError('Invalid data directory: %s' % args['DATA_DIR'])

 counts = wordcount(glob.glob(os.path.join(args['DATA_DIR'], '*.txt')))
 logging.debug(counts)
```

### 17.7.3 Serial Implementation Using map and reduce

We can improve the serial implementation in anticipation of parallelizing the program by making use of Python's `map` and `reduce` functions.

In short, you can use `map` to apply the same function to the members of a collection. For example, to convert a list of numbers to strings, you could do:

```
import random
nums = [random.randint(1, 2) for _ in range(10)]
print(nums)
[2, 1, 1, 1, 2, 2, 2, 2, 2]
print(map(str, nums))
['2', '1', '1', '1', '2', '2', '2', '2', '2']
```

We can use `reduce` to apply the same function cumulatively to the items of a sequence. For example, to find the total of the numbers in our list, we could use `reduce` as follows:

```
def add(x, y):
 return x + y

print(reduce(add, nums))
```

We can simplify this even more by using a lambda function:

```
print(reduce(lambda x, y: x + y, nums))
```

You can read more about [Python's lambda function in the docs](#).

With this in mind, we can reimplement the `wordcount` example as follows:

```

'''Usage: wordcount_mapreduce.py [-h] DATA_DIR

Read a collection of .txt documents and count how
many times each word
appears in the collection.

Arguments:
 DATA_DIR A directory with documents (.txt files).

Options:
 -h --help
 ...

from __future__ import division, print_function
import os, glob, logging
from docopt import docopt

logging.basicConfig(level=logging.DEBUG)

def count_words(filepath):
 counts = {}
 with open(filepath, 'r') as f:
 words = [word.strip() for word in f.read().split()]

 for word in words:
 if word not in counts:
 counts[word] = 0
 counts[word] += 1
 return counts

def merge_counts(counts1, counts2):
 for word, count in counts2.items():
 if word not in counts1:
 counts1[word] = 0
 counts1[word] += counts2[word]
 return counts1

if __name__ == '__main__':
 args = docopt(__doc__)
 if not os.path.exists(args['DATA_DIR']):
 raise ValueError('Invalid data directory: %s' % args['DATA_DIR'])

 per_doc_counts = map(count_words,
 glob.glob(os.path.join(args['DATA_DIR'],
 '*.txt')))
 counts = reduce(merge_counts, [{}] + per_doc_counts)
 logging.debug(counts)

```

## 17.7.4 Parallel Implementation

Drawing on the previous implementation using `map` and `reduce`, we can parallelize the implementation using Python's `multiprocessing` API:

```

'''Usage: wordcount_mapreduce_parallel.py [-h] DATA_DIR NUM_PROCESSES

Read a collection of .txt documents and count, in parallel, how many
times each word appears in the collection.

Arguments:
 DATA_DIR A directory with documents (.txt files).
 NUM_PROCESSES The number of parallel processes to use.

Options:
 -h --help
 ...

from __future__ import division, print_function
import os, glob, logging
from docopt import docopt
from wordcount_mapreduce import count_words, merge_counts
from multiprocessing import Pool

logging.basicConfig(level=logging.DEBUG)

if __name__ == '__main__':
 args = docopt(__doc__)
 if not os.path.exists(args['DATA_DIR']):
 raise ValueError('Invalid data directory: %s' % args['DATA_DIR'])
 num_processes = int(args['NUM_PROCESSES'])

 pool = Pool(processes=num_processes)

 per_doc_counts = pool.map(count_words,
 glob.glob(os.path.join(args['DATA_DIR'],
 '*.txt')))
 counts = reduce(merge_counts, [{}] + per_doc_counts)
 logging.debug(counts)

```

## 17.7.5 Benchmarking

To time each of the examples, enter it into its own Python file and use Linux's `time` command:

```
$ time python wordcount.py docs-1000-10000
```

The output contains the real run time and the user run time. `real` is wall clock time - time from start to finish of the call. `user` is the amount of CPU time spent in user-mode code (outside the kernel) within the process, that is, only actual CPU time used in executing the process.

## 17.7.6 Exercises

## E.python.wordcount.1:

Run the three different programs (serial, serial w/ map and reduce, parallel) and answer the following questions:

1. Is there any performance difference between the different versions of the program?
2. Does user time significantly differ from real time for any of the versions of the program?
3. Experiment with different numbers of processes for the parallel example, starting with 1. What is the performance gain when you go from 1 to 2 processes? From 2 to 3? When do you stop seeing improvement? (this will depend on your machine architecture)

## 17.7.7 References

- [Map, Filter and Reduce](#)
- [multiprocessing API](#)

## 17.8 NumPy



NumPy is a popular library that is used by many other Python libraries such as pandas, and SciPy. It provides simple to use array operations for data. This helps to access arrays in a more intuitive fashion and introduces various matrix operations.

We provide a short introduction to NumPy.

First we import the modules needed for this introduction and abbreviate them with the `as` feature of the import statement

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Now we showcase some features of NumPy.

## 17.8.1 Float Range

`arange()` is like `range()`, but for floating-point numbers.

```
X = np.arange(0.2, 1, .1)
print (X)
```

We use this function to generate parameter space that can then be iterated on.

```
P = 10.0 ** np.arange(-7, 1, 1)
print (P)

for x,p in zip(X,P):
 print ('%f, %f' % (x, p))
```

## 17.8.2 Arrays

To create one dimensional arrays you use

```
a = np.array([1, 2, 3])
```

To check some properties you can use the following

```
print(type(a)) # Prints "<class 'numpy.ndarray'>"
print(a.shape) # Prints "(3,)"
```

The shape indicates that in the first dimension, there are 3 elements.  
To print the actual values you can use

```
print(a) # Prints the values of the array
print(a[0], a[1], a[2]) # Prints "1 2 3"
```

To change values you can use the index of the element or use any other python method to do so. IN our example we change the first element to 42

```
a[0] = 42
print(a)
```

To create more dimensional arrays you use

```
b = np.array([[1,2,3],[4,5,6]]) # Create a 2 dimensional array
print(b.shape) # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0]) # Prints "1 2 4"
```

### 17.8.3 Array Operations

Let us first create some arrays with a predefined datatype

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

print (x)

print(y)
```

To add the numbers use

```
print(x+y)
```

Other functions such as `-`, `*`, `/` behave as expected using elementwise operations:

```
print(x-y)

print(x*y)

print(x/y)
```

To apply functions such as 'sin' make sure you use the function provided by the numpy package such as `np.sin`. The list of functions is included in the manual at \* <https://docs.scipy.org/doc/numpy/reference/routines.math.html>

```
print (np.sin(x))

print (np.sum(x))
```

Computations can also be applied to columns and rows

```
print(np.sum(x, axis=0)) # sum of each column
print(np.sum(x, axis=1)) # sum of each row
```

### 17.8.4 Linear Algebra

Linear algebra methods are also provided.

```
from numpy import linalg

A = np.diag((1,2,3))

w,v = linalg.eig(A)

print ('w =', w)
print ('v =', v)
```

## 17.8.5 Resources

- <https://docs.scipy.org/doc/numpy/>
- <http://cs231n.github.io/python-numpy-tutorial/#numpy>

## 17.9 SciPy



SciPy is a library built around numpy and has a number of off-the-shelf algorithms and operations implemented. These include algorithms from calculus (such as integration), statistics, linear algebra, image-processing, signal processing, machine learning.

To achieve this, SciPy bundles a number of useful open-source software for mathematics, science, and engineering. It includes the following packages:

NumPy,

for managing N-dimensional arrays

SciPy library,

to access fundamental scientific computing capabilities

Matplotlib,

to conduct 2D plotting

IPython,

for an Interactive console (see jupyter)

Sympy,  
for symbolic mathematics  
pandas,  
for providing data structures and analysis

### 17.9.1 Introduction

First we add the usual scientific computing modules with the typical abbreviations, including sp for scipy. We could invoke scipy's statistical package as sp.stats, but for the sake of laziness we abbreviate that too.

```
import numpy as np # import numpy
import scipy as sp # import scipy
from scipy import stats # refer directly to stats rather than sp.stats
import matplotlib as mpl # for visualization
from matplotlib import pyplot as plt # refer directly to pyplot
 # rather than mpl.pyplot
```

Now we create some random data to play with. We generate 100 samples from a Gaussian distribution centered at zero.

```
s = sp.randn(100)
```

How many elements are in the set?

```
print ('There are',len(s), 'elements in the set')
```

What is the mean (average) of the set?

```
print ('The mean of the set is',s.mean())
```

What is the minimum of the set?

```
print ('The minimum of the set is',s.min())
```

What is the maximum of the set?

```
print ('The maximum of the set is',s.max())
```

We can use the scipy functions too. What's the median?

```
print ('The median of the set is',sp.median(s))
```

What about the standard deviation and variance?

```
print ('The standard deviation is',sp.std(s),
 'and the variance is',sp.var(s))
```

Isn't the variance the square of the standard deviation?

```
print ('The square of the standard deviation is',sp.std(s)**2)
```

How close are the measures? The differences are close as the following calculation shows

```
print ('The difference is',abs(sp.std(s)**2 - sp.var(s)))

print ('And in decimal form, the difference is %.16f' %
 (abs(sp.std(s)**2 - sp.var(s))))
```

How does this look as a histogram? See [Figure 163](#), [Figure 164](#), [Figure 165](#)

```
plt.hist(s) # yes, one line of code for a histogram
plt.show()
```

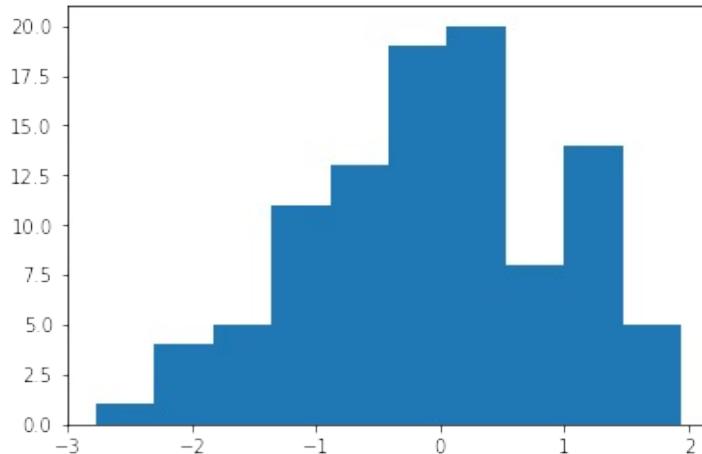


Figure 163: Histogram 1

Let's add some titles.

```
plt.clf() # clear out the previous plot

plt.hist(s)
plt.title("Histogram Example")
plt.xlabel("Value")
```

```
plt.ylabel("Frequency")
plt.show()
```

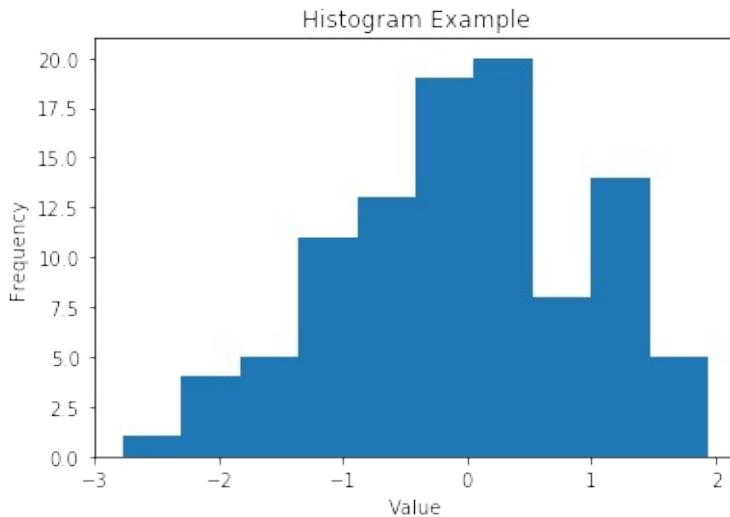


Figure 164: Histogram 2

Typically we do not include titles when we prepare images for inclusion in LaTeX. There we use the caption to describe what the figure is about.

```
plt.clf() # clear out the previous plot

plt.hist(s)
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.show()
```

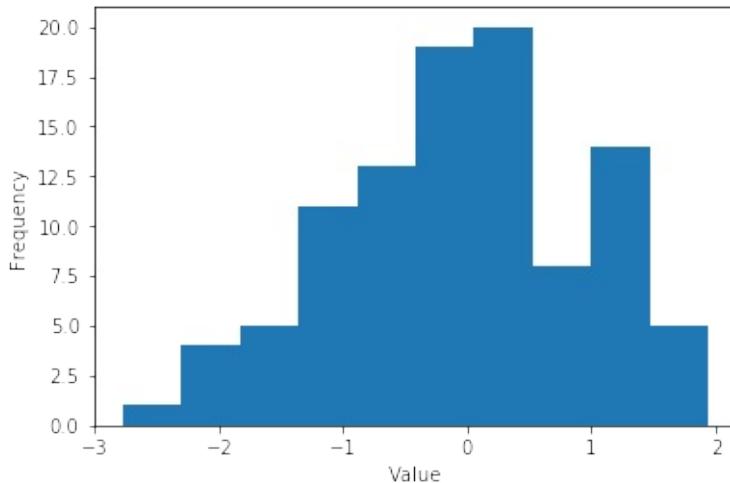


Figure 165: Histogram 3

Let's try out some linear regression, or curve fitting. See @#fig:scipy-output\_30\_0

```
import random

def F(x):
 return 2*x - 2

def add_noise(x):
 return x + random.uniform(-1,1)

X = range(0,10,1)

Y = []
for i in range(len(X)):
 Y.append(add_noise(X[i]))

plt.clf() # clear out the old figure
plt.plot(X,Y, '.')
plt.show()
```

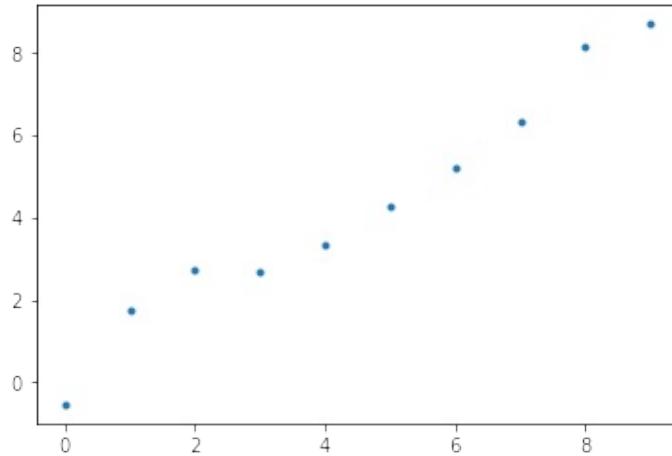


Figure 166: Result 1

Now let's try linear regression to fit the curve.

```
m, b, r, p, est_std_err = stats.linregress(X,Y)
```

What is the slope and y-intercept of the fitted curve?

```
print ('The slope is',m,'and the y-intercept is', b)

def Fprime(x): # the fitted curve
 return m*x + b
```

Now let's see how well the curve fits the data. We'll call the fitted curve F'.

```
x = range(0,10,1)

Yprime = []
for i in range(len(X)):
 Yprime.append(Fprime(X[i]))

plt.clf() # clear out the old figure

the observed points, blue dots
plt.plot(X, Y, '.', label='observed points')

the interpolated curve, connected red line
plt.plot(X, Yprime, 'r-', label='estimated points')

plt.title("Linear Regression Example") # title
plt.xlabel("x") # horizontal axis title
plt.ylabel("y") # vertical axis title
legend labels to plot
plt.legend(['observed points', 'estimated points'])

comment out so that you can save the figure
plt.show()
```

To save images into a PDF file for inclusion into LaTeX documents you can save the images as follows. Other formats such as png are also possible, but the quality is naturally not sufficient for inclusion in papers and documents. For that you certainly want to use PDF. The save of the figure has to occur before you use the `show()` command. See [Figure 167](#)

```
plt.savefig("regression.pdf", bbox_inches='tight')

plt.savefig('regression.png')

plt.show()
```

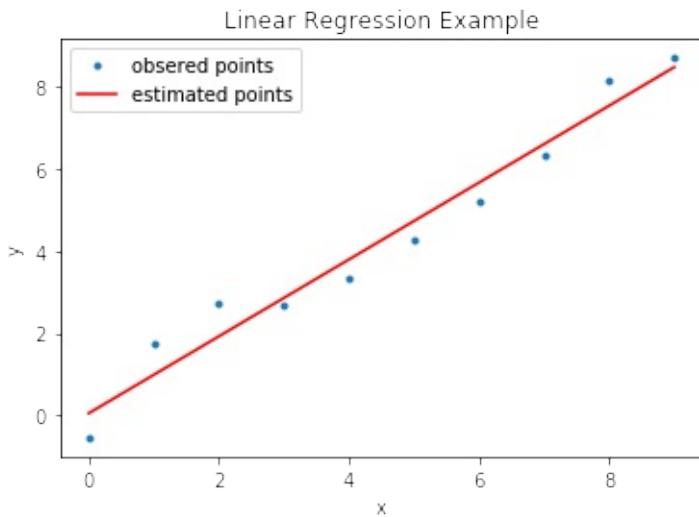


Figure 167: Result 2

## 17.9.2 References

For more information about SciPy we recommend that you visit the following link

<https://www.scipy.org/getting-started.html#learning-to-work-with-scipy>

Additional material and inspiration for this section are from

- [O] “Getting Started guide” <https://www.scipy.org/getting-started.html>
- [O] Prasanth. “Simple statistics with SciPy.” Comfort at 1 AU. February 28, 2011. <https://oneau.wordpress.com/2011/02/28/simple-statistics-with-scipy/>.
- [O] SciPy Cookbook. Lasted updated: 2015. <http://scipy-cookbook.readthedocs.io/>.

O create bibtex entries

## 17.10 SCIKIT-LEARN O



In this section we demonstrate how simple it is to use k-means in

scikit learn.

### 17.10.1 Instalation

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using pip

```
$ pip install numpy
$ pip install scipy -U
$ pip install -U scikit-learn
```

### 17.10.2 Import

```
from time import time
import numpy as np
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
```

### 17.10.3 Create samples

```
np.random.seed(42)

digits = load_digits()
data = scale(digits.data)

n_samples, n_features = data.shape
n_digits = len(np.unique(digits.target))
labels = digits.target

sample_size = 300

print("n_digits: %d, \t n_samples %d, \t n_features %d"
 % (n_digits, n_samples, n_features))

print(79 * '_')
print('% 9s' % 'init'
 ' time inertia homo compl v-meas ARI AMI silhouette')

def bench_k_means(estimator, name, data):
 t0 = time()
 estimator.fit(data)
 print('% 9s %.2fs %i %.3f %.3f %.3f %.3f %.3f'
 % (name, (time() - t0), estimator.inertia_,
```

```

metrics.homogeneity_score(labels, estimator.labels_),
metrics.completeness_score(labels, estimator.labels_),
metrics.v_measure_score(labels, estimator.labels_),
metrics.adjusted_rand_score(labels, estimator.labels_),
metrics.adjusted_mutual_info_score(labels, estimator.labels_),
metrics.silhouette_score(data, estimator.labels_,
 metric='euclidean',
 sample_size=sample_size))

bench_k_means(KMeans(init='k-means++', n_clusters=n_digits, n_init=10),
 name="k-means++", data=data)

bench_k_means(KMeans(init='random', n_clusters=n_digits, n_init=10),
 name="random", data=data)

in this case the seeding of the centers is deterministic, hence we run the
kmeans algorithm only once with n_init=1
pca = PCA(n_components=n_digits).fit(data)
bench_k_means(KMeans(init=pca.components_,
 n_clusters=n_digits, n_init=1),
 name="PCA-based",
 data=data)
print(79 * '_')

```

## 17.11 VISUALIZE

---

See [Figure 168](#)

```

reduced_data = PCA(n_components=2).fit_transform(data)
kmeans = KMeans(init='k-means++', n_clusters=n_digits, n_init=10)
kmeans.fit(reduced_data)

Step size of the mesh. Decrease to increase the quality of the VQ.
h = .02 # point in the mesh [x_min, x_max]x[y_min, y_max].

Plot the decision boundary. For that, we will assign a color to each
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

Obtain labels for each point in mesh. Use last trained model.
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])

Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
 extent=(xx.min(), xx.max(), yy.min(), yy.max()),
 cmap=plt.cm.Paired,
 aspect='auto', origin='lower')

plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
Plot the centroids as a white X
centroids = kmeans.cluster_centers_

```

```

plt.scatter(centroids[:, 0], centroids[:, 1],
 marker='x', s=169, linewidths=3,
 color='w', zorder=10)
plt.title('K-means clustering on the digits dataset (PCA-reduced data)\n'
 'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

```

K-means clustering on the digits dataset (PCA-reduced data)  
 Centroids are marked with white cross

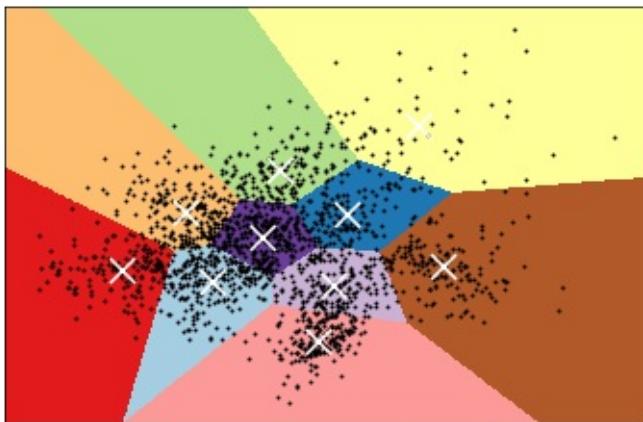


Figure 168: Result

## 17.12 PARALLEL COMPUTING IN PYTHON



In this module we will review the available Python modules that can be used for parallel computing. The parallel computing can be in form of either multi-threading or multi-processing. In multi-threading approach, the threads run in the same shared memory heap whereas in case of multi-processing, the memory heaps of processes are separate and independent, therefore the communication between the processes are a little bit more complex.

### 17.12.1 Multi-threading in Python

Threading in Python is perfect for I/O operations where the process is expected to be idle regularly, e.g. web scraping. This is a very useful feature because several applications and script might spend the majority of their runtime on waiting for network or data I/O. In

several cases, e.g. web scraping, the resources, i.e. downloading from different websites, are most of the time independent. Therefore the processor can download in parallel and join the result at the end.

### 17.12.1.1 Thread **vs** Threading

There are two built-in modules in Python that are related to threading, namely `thread` and `threading`. The former module is deprecated for sometime in Python 2, and in Python 3 it is renamed to `_thread` for the sake of backwards incompatibilities. The `_thread` module provides low-level threading API for multi-threading in Python, whereas the module `threading` builds a high-level threading interface on top of it.

The `Thread()` is the main method of the `threading` module, the two important arguments of which are `target`, for specifying the callable object, and `args` to pass the arguments for the target callable. We illustrate these in the following example:

```
import threading

def hello_thread(thread_num):
 print ("Hello from Thread ", thread_num)

if __name__ == '__main__':
 for thread_num in range(5):
 t = threading.Thread(target=hello_thread, args=(thread_num,))
 t.start()
```

This is the output of the previous example:

```
In [1]: %run threading.py
Hello from Thread 0
Hello from Thread 1
Hello from Thread 2
Hello from Thread 3
Hello from Thread 4
```

In case you are not familiar with the `if __name__ == '__main__':` statement, what it does is basically making sure that the code nested under this condition will be run only if you run your module as a program and it will not run in case your module is imported in another file.

### 17.12.1.2 Locks

As mentioned prior, the memory space is shared between the threads. This is at the same time beneficial and problematic: it is beneficial in a sense that the communication between the threads becomes easy, however, you might experience strange outcome if you let several threads change same variable without caution, e.g. thread 2 changes variable `x` while thread 1 is working with it. This is when `lock` comes into play. Using `lock`, you can allow only one thread to work with a variable. In other words, only a single thread can hold the `lock`. If the other threads need to work with that variable, they have to wait until the other thread is done and the variable is “unlocked”.

We illustrate this with a simple example:

```
import threading

global counter
counter = 0

def incrementer1():
 global counter
 for j in range(2):
 for i in range(3):
 counter += 1
 print("Greeter 1 incremented the counter by 1")
 print ("Counter is %d"%counter)

def incrementer2():
 global counter
 for j in range(2):
 for i in range(3):
 counter += 1
 print("Greeter 2 incremented the counter by 1")
 print ("Counter is now %d"%counter)

if __name__ == '__main__':
 t1 = threading.Thread(target = incrementer1)
 t2 = threading.Thread(target = incrementer2)

 t1.start()
 t2.start()
```

Suppose we want to print multiples of 3 between 1 and 12, i.e. 3, 6, 9 and 12. For the sake of argument, we try to do this using 2 threads

and a nested for loop. Then we create a global variable called counter and we initialize it with 0. Then whenever each of the `incrementer1` or `incrementer2` functions are called, the `counter` is incremented by 3 twice (counter is incremented by 6 in each function call). If you run the previous code, you should be really lucky if you get the following as part of your output:

bash Counter is now 3 Counter is now 6 Counter is now 9 Counter is now 12 The reason is the conflict that happens between threads while incrementing the `counter` in the nested for loop. As you probably noticed, the first level for loop is equivalent of adding 3 to the counter and the conflict that might happen is not effective on that level but the nested for loop. Accordingly, the output of the previous code is different in every run. This is an example output:

```
$ python3 lock_example.py
Greeter 1 incremented the counter by 1
Greeter 1 incremented the counter by 1
Greeter 1 incremented the counter by 1
Counter is 4
Greeter 2 incremented the counter by 1
Greeter 2 incremented the counter by 1
Greeter 1 incremented the counter by 1
Greeter 2 incremented the counter by 1
Greeter 1 incremented the counter by 1
Counter is 8
Greeter 1 incremented the counter by 1
Greeter 2 incremented the counter by 1
Counter is 10
Greeter 2 incremented the counter by 1
Greeter 2 incremented the counter by 1
Counter is 12
```

We can fix this issue using a `lock`: whenever one of the function is going to increment the value by 3, it will `acquire()` the lock and when it is done the function will `release()` the lock. This mechanism is illustrated in the following code:

```
import threading

increment_by_3_lock = threading.Lock()

global counter
counter = 0

def incrementer1():
 global counter
```

```

for j in range(2):
 increment_by_3_lock.acquire(True)
 for i in range(3):
 counter += 1
 print("Greeter 1 incremented the counter by 1")
 print ("Counter is %d"%counter)
 increment_by_3_lock.release()

def incrementer2():
 global counter
 for j in range(2):
 increment_by_3_lock.acquire(True)
 for i in range(3):
 counter += 1
 print("Greeter 2 incremented the counter by 1")
 print ("Counter is %d"%counter)
 increment_by_3_lock.release()

if __name__ == '__main__':
 t1 = threading.Thread(target = incrementer1)
 t2 = threading.Thread(target = incrementer2)

 t1.start()
 t2.start()

```

No matter how many times you run this code, the output would always be in the correct order:

```

$ python3 lock_example.py
Greeter 1 incremented the counter by 1
Greeter 1 incremented the counter by 1
Greeter 1 incremented the counter by 1
Counter is 3
Greeter 1 incremented the counter by 1
Greeter 1 incremented the counter by 1
Greeter 1 incremented the counter by 1
Counter is 6
Greeter 2 incremented the counter by 1
Greeter 2 incremented the counter by 1
Greeter 2 incremented the counter by 1
Counter is 9
Greeter 2 incremented the counter by 1
Greeter 2 incremented the counter by 1
Greeter 2 incremented the counter by 1
Counter is 12

```

Using the `Threading` module increases both the overhead associated with thread management as well as the complexity of the program and that is why in many situations, employing `multiprocessing` module might be a better approach.

## 17.12.2 Multi-processing in Python

We already mentioned that multi-threading might not be sufficient in many applications and we might need to use `multiprocessing` sometime, or better to say most of the times. That is why we are dedicating this subsection to this particular module. This module provides you with an API for spawning processes the way you spawn threads using `threading` module. Moreover, there are some functionalities that are not even available in `threading` module, e.g. the `Pool` class which allows you to run a batch of jobs using a pool of worker processes.

### 17.12.2.1 Process

Similar to `threading` module which was employing `thread` (aka `_thread`) under the hood, `multiprocessing` employs the `Process` class. Consider the following example:

```
from multiprocessing import Process
import os

def greeter(name):
 proc_idx = os.getpid()
 print("Process {0}: Hello {1}!".format(proc_idx, name))

if __name__ == '__main__':
 name_list = ['Harry', 'George', 'Dirk', 'David']
 process_list = []
 for name_idx, name in enumerate(name_list):
 current_process = Process(target=greeter, args=(name,))
 process_list.append(current_process)
 current_process.start()
 for process in process_list:
 process.join()
```

In this example, after importing the `Process` module we created a `greeter()` function that takes a `name` and greets that person. It also prints the `pid` (process identifier) of the process that is running it. Note that we used the `os` module to get the `pid`. In the bottom of the code after checking the `__name__='__main__'` condition, we create a series of `Processes` and `start` them. Finally in the last for loop and using the `join` method, we tell Python to wait for the processes to terminate. This is one of the possible outputs of the code:

```
$ python3 process_example.py
Process 23451: Hello Harry!
Process 23452: Hello George!
```

```
Process 23453: Hello Dirk!
Process 23454: Hello David!
```

## 17.12.2.2 Pool

Consider the `Pool` class as a pool of worker processes. There are several ways for assigning jobs to the `Pool` class and we will introduce the most important ones in this section. These methods are categorized as `blocking` or `non-blocking`. The former means that after calling the API, it blocks the thread/process until it has the result or answer ready and the control returns only when the call completes. In the `non-blocking` on the other hand, the control returns immediately.

### 17.12.2.2.1 Synchronous `Pool.map()`

We illustrate the `Pool.map` method by re-implementing our previous greeter example using `Pool.map`:

```
`python from multiprocessing import Pool import os

def greeter(name): pid = os.getpid() print("Process {0}: Hello {1}!".format(pid,name))

if __name__ == '__main__': names = ['Jenna', 'David','Marry', 'Ted','Jerry','Tom','Justin'] pool = Pool(processes=3) sync_map = pool.map(greeter,names) print("Done!")`
```

As you can see, we have seven names here but we do not want to dedicate each greeting to a separate process. Instead we do the whole job of “greeting seven people” using “two processes”. We create a pool of 3 processes with `Pool(processes=3)` syntax and then we map an iterable called `names` to the `greeter` function using `pool.map(greeter,names)`. As we expected, the greetings in the output will be printed from three different processes:

```
$ python poolmap_example.py
Process 30585: Hello Jenna!
Process 30586: Hello David!
Process 30587: Hello Marry!
Process 30585: Hello Ted!
Process 30585: Hello Jerry!
```

```
Process 30587: Hello Tom!
Process 30585: Hello Justin!
Done!
```

Note that `Pool.map()` is in `blocking` category and does not return the control to your script until it is done calculating the results. That is why `Done!` is printed after all of the greetings are over.

#### 17.12.2.2 Asynchronous `Pool.map_async()`

As the name implies, you can use the `map_async` method, when you want assign many function calls to a pool of worker processes asynchronously. Note that unlike `map`, the order of the results is not guaranteed (as oppose to `map`) and the control is returned immediately. We now implement the previous example using `map_async`:

```
from multiprocessing import Pool
import os

def greeter(name):
 pid = os.getpid()
 print("Process {0}: Hello {1}!".format(pid, name))

if __name__ == '__main__':
 names = ['Jenna', 'David', 'Marry', 'Ted', 'Jerry', 'Tom', 'Justin']
 pool = Pool(processes=3)
 async_map = pool.map_async(greeter, names)
 print("Done!")
 async_map.wait()
```

As you probably noticed, the only difference (clearly apart from the `map_async` method name) is calling the `wait()` method in the last line. The `wait()` method tells your script to wait for the result of `map_async` before terminating:

```
$ python poolmap_example.py
Done!
Process 30740: Hello Jenna!
Process 30741: Hello David!
Process 30740: Hello Ted!
Process 30742: Hello Marry!
Process 30740: Hello Jerry!
Process 30741: Hello Tom!
Process 30742: Hello Justin!
```

Note that the order of the results are not preserved. Moreover, `Done!` is printer before any of the results, meaning that if we do not use the

`wait()` method, you probably will not see the result at all.

### 17.12.2.3 Locks

The way `multiprocessing` module implements locks is almost identical to the way the `threading` module does. After importing `Lock` from `multiprocessing` all you need to do is to `acquire` it, do some computation and then `release` the lock. We will clarify the use of `Lock` by providing an example in next section about process communication.

### 17.12.2.4 Process Communication

Process communication in `multiprocessing` is one of the most important, yet complicated, features for better use of this module. As oppose to `threading`, the `Process` objects will not have access to any shared variable by default, i.e. no shared memory space between the processes by default. This effect is illustrated in the following example:

```
from multiprocessing import Process, Lock, Value
import time

global counter
counter = 0

def incrementer1():
 global counter
 for j in range(2):
 for i in range(3):
 counter += 1
 print ("Greeter1: Counter is %d"%counter)

def incrementer2():
 global counter
 for j in range(2):
 for i in range(3):
 counter += 1
 print ("Greeter2: Counter is %d"%counter)

if __name__ == '__main__':
 t1 = Process(target = incrementer1)
 t2 = Process(target = incrementer2)
 t1.start()
 t2.start()
```

Probably you already noticed that this is almost identical to our example in `threading` section. Now, take a look at the strange output:

```
$ python communication_example.py
Greeter1: Counter is 3
Greeter1: Counter is 6
Greeter2: Counter is 3
Greeter2: Counter is 6
```

As you can see, it is as if the processes does not see each other. Instead of having two processes one counting to 6 and the other counting from 6 to 12, we have two processes counting to 6.

Nevertheless, there are several ways that `Process`s from `multiprocessing` can communicate with each other, including `Pipe`, `Queue`, `Value`, `Array` and `Manager`. `Pipe` and `Queue` are appropriate for inter-process message passing. To be more specific, `Pipe` is useful for process-to-process scenarios while `Queue` is more appropriate for `processes`-toprocesses ones. `Value` and `Array` are both used to provide a synchronized access to a shared data (very much like shared memory) and `Managers` can be used on different data types. In the following sub-sections, we cover both `Value` and `Array` since they are both lightweight, yet useful, approaches.

#### 17.12.2.4.1 Value

The following example re-implements the broken example in the previous section. We fix the strange output, by using both `Lock` and `value`:

```
from multiprocessing import Process, Lock, Value
import time

increment_by_3_lock = Lock()

def incrementer1(counter):
 for j in range(3):
 increment_by_3_lock.acquire(True)
 for i in range(3):
 counter.value += 1
 time.sleep(0.1)
 print ("Greeter1: Counter is %d"%counter.value)
 increment_by_3_lock.release()
```

```

def incrementer2(counter):
 for j in range(3):
 increment_by_3_lock.acquire(True)
 for i in range(3):
 counter.value += 1
 time.sleep(0.05)
 print ("Greeter2: Counter is %d"%counter.value)
 increment_by_3_lock.release()

if __name__ == '__main__':
 counter = Value('i',0)
 t1 = Process(target = incrementer1, args=(counter,))
 t2 = Process(target = incrementer2 , args=(counter,))
 t2.start()
 t1.start()

```

The usage of `Lock` object in this example is identical to the example in `threading` section. The usage of `counter` is on the other hand the novel part. First, note that `counter` is not a global variable anymore and instead it is a `value` which returns a `ctypes` object allocated from a shared memory between the processes. The first argument `'i'` indicates a signed integer, and the second argument defines the initialization value. In this case we are assigning a signed integer in the shared memory initialized to size 0 to the `counter` variable. We then modified our two functions and pass this shared variable as an argument. Finally, we change the way we increment the `counter` since `counter` is not an Python integer anymore but a `ctypes` signed integer where we can access its value using the `value` attribute. The output of the code is now as we expected:

```

$ python mp_lock_example.py
Greeter2: Counter is 3
Greeter2: Counter is 6
Greeter1: Counter is 9
Greeter1: Counter is 12

```

The last example related to parallel processing, illustrates the use of both `value` and `Array`, as well as a technique to pass multiple arguments to a function. Note that the `Process` object does not accept multiple arguments for a function and therefore we need this or similar techniques for passing multiple arguments. Also, this technique can also be used when you want to pass multiple arguments to `map` or `map_async`:

```

from multiprocessing import Process, Lock, Value, Array
import time
from ctypes import c_char_p

increment_by_3_lock = Lock()

def incrementer1(counter_and_names):
 counter= counter_and_names[0]
 names = counter_and_names[1]
 for j in range(2):
 increment_by_3_lock.acquire(True)
 for i in range(3):
 counter.value += 1
 time.sleep(0.1)
 name_idx = counter.value//3 -1
 print ("Greeter1: Greeting {0}! Counter is {1}".format(names.value[name_idx],counter.value))
 increment_by_3_lock.release()

def incrementer2(counter_and_names):
 counter= counter_and_names[0]
 names = counter_and_names[1]
 for j in range(2):
 increment_by_3_lock.acquire(True)
 for i in range(3):
 counter.value += 1
 time.sleep(0.05)
 name_idx = counter.value//3 -1
 print ("Greeter2: Greeting {0}! Counter is {1}".format(names.value[name_idx],counter.value))
 increment_by_3_lock.release()

if __name__ == '__main__':
 counter = Value('i',0)
 names = Array (c_char_p,4)
 names.value = ['James','Tom','Sam', 'Larry']
 t1 = Process(target = incrementer1, args=((counter,names),))
 t2 = Process(target = incrementer2 , args=((counter,names),))
 t2.start()
 t1.start()

```

In this example we created a `multiprocessing.Array()` object and assigned it to a variable called `names`. As we mentioned before, the first argument is the `ctype` data type and since we want to create an array of strings with length of 4 (second argument), we imported the `c_char_p` and passed it as the first argument.

Instead of passing the arguments separately, we merged both the `value` and `Array` objects in a tuple and passed the tuple to the functions. We then modified the functions to unpack the objects in the first two

lines in the both functions. Finally we changed the print statement in a way that each process greets a particular name. The output of the example is:

```
$ python3 mp_lock_example.py
Greeter2: Greeting James! Counter is 3
Greeter2: Greeting Tom! Counter is 6
Greeter1: Greeting Sam! Counter is 9
Greeter1: Greeting Larry! Counter is 12
```

## 17.13 Dask



Dask is a python-based parallel computing library for analytics. Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved concurrently.

Dask is composed of two components:

1. Dynamic task scheduling optimized for computation. This is similar to Airflow, Luigi, Celery, or Make, but optimized for interactive computational workloads.
2. Big Data collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments. These parallel collections run on top of the dynamic task schedulers.

Dask emphasizes the following virtues:

- Familiar: Provides parallelized NumPy array and Pandas DataFrame objects.
- Flexible: Provides a task scheduling interface for more custom workloads and integration with other projects.
- Native: Enables distributed computing in Pure Python with access to the PyData stack.
- Fast: Operates with low overhead, low latency, and minimal serialization necessary for fast numerical algorithms

- Scales up: Runs resiliently on clusters with 1000s of cores
- Scales down: Trivial to set up and run on a laptop in a single process
- Responsive: Designed with interactive computing in mind it provides rapid feedback and diagnostics to aid humans

The section is structured in a number of subsections addressing the following topics:

Foundations:

an explanation of what Dask is, how it works, and how to use lower level primitives to set up computations. Casual users may wish to skip this section, although we consider it useful knowledge for all users.

Distributed Features:

information on running Dask on the distributed scheduler, which enables scale-up to distributed settings and enhanced monitoring of task operations. The distributed scheduler is now generally the recommended engine for executing task work, even on single workstations or laptops.

Collections:

convenient abstractions giving a familiar feel to big data.

Bags:

Python iterators with a functional paradigm, such as found in func/iter-tools and toolz - generalize lists/generators to big data; this will seem very familiar to users of PySpark's RDD

Array:

massive multi-dimensional numerical data, with Numpy functionality

Dataframe:

massive tabular data, with Pandas functionality

### 17.13.1 How Dask Works

Dask is computation tool for larger-than-memory datasets, parallel execution or delayed/background execution.

We can summarize the basics of Dask as follows:

- process data that does not fit into memory by breaking it into blocks and specifying task chains
- parallelize execution of tasks across cores and even nodes of a cluster
- move computation to the data rather than the other way around, to minimize communication overheads

We use for-loops to build basic tasks, Python iterators, and the Numpy (array) and Pandas (dataframe) functions for multi-dimensional or tabular data, respectively.

Dask allows us to construct a prescription for the calculation we want to carry out. A module named Dask.delayed lets us parallelize custom code. It is useful whenever our problem doesn't quite fit a high-level parallel object like dask.array or dask.dataframe but could still benefit from parallelism. Dask.delayed works by delaying our function evaluations and putting them into a dask graph. Here is a small example:

```
from dask import delayed

@delayed
def inc(x):
 return x + 1

@delayed
def add(x, y):
 return x + y
```

Here we have used the delayed annotation to show that we want

these functions to operate lazily - to save the set of inputs and execute only on demand.

## 17.13.2 Dask Bag

Dask-bag excels in processing data that can be represented as a sequence of arbitrary inputs. We'll refer to this as "messy" data, because it can contain complex nested structures, missing fields, mixtures of data types, etc. The functional programming style fits very nicely with standard Python iteration, such as can be found in the `itertools` module.

Messy data is often encountered at the beginning of data processing pipelines when large volumes of raw data are first consumed. The initial set of data might be JSON, CSV, XML, or any other format that does not enforce strict structure and datatypes. For this reason, the initial data massaging and processing is often done with Python lists, dicts, and sets.

These core data structures are optimized for general-purpose storage and processing. Adding streaming computation with iterators/generator expressions or libraries like `itertools` or `toolz` let us process large volumes in a small space. If we combine this with parallel processing then we can churn through a fair amount of data.

Dask.bag is a high level Dask collection to automate common workloads of this form. In a nutshell

```
dask.bag = map, filter, toolz + parallel execution
```

You can create a Bag from a Python sequence, from files, from data on S3, etc..

```
each element is an integer
import dask.bag as db
b = db.from_sequence([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

each element is a text file of JSON lines
import os
b = db.read_text(os.path.join('data', 'accounts.*.json.gz'))
```

```
Requires `s3fs` library
each element is a remote CSV text file
b = db.read_text('s3://dask-data/nyc-taxi/2015/yellow_tripdata_2015-01.csv')
```

Bag objects hold the standard functional API found in projects like the Python standard library, toolz, or pyspark, including map, filter, groupby, etc..

As with Array and DataFrame objects, operations on Bag objects create new bags. Call the .compute() method to trigger execution.

```
def is_even(n):
 return n % 2 == 0

b = db.from_sequence([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
c = b.filter(is_even).map(lambda x: x ** 2)
c

blocking form: wait for completion (which is very fast in this case)
c.compute()
```

For more details on Dask Bag check <https://dask.pydata.org/en/latest/bag.html>

### 17.13.3 Concurrency Features

Dask supports a real-time task framework that extends Python's concurrent.futures interface. This interface is good for arbitrary task scheduling, like dask.delayed, but is immediate rather than lazy, which provides some more flexibility in situations where the computations may evolve over time. These features depend on the second generation task scheduler found in dask.distributed (which, despite its name, runs very well on a single machine).

Dask allows us to simply construct graphs of tasks with dependencies. We can find that graphs can also be created automatically for us using functional, Numpy or Pandas syntax on data collections. None of this would be very useful, if there weren't also a way to execute these graphs, in a parallel and memory-aware way. Dask comes with four available schedulers:

- `dask.threaded.get`: a scheduler backed by a thread pool

- `dask.multiprocessing.get`: a scheduler backed by a process pool
- `dask.async.get_sync`: a synchronous scheduler, good for debugging
- `distributed.Client.get`: a distributed scheduler for executing graphs on multiple machines.

Here is a simple program for `dask.distributed` library:

```
from dask.distributed import Client
client = Client('scheduler:port')

futures = []
for fn in filenames:
 future = client.submit(load, fn)
 futures.append(future)

summary = client.submit(summarize, futures)
summary.result()
```

For more details on Concurrent Features by Dask check  
<https://dask.pydata.org/en/latest/futures.html>

### 17.13.4 Dask Array

Dask arrays implement a subset of the NumPy interface on large arrays using blocked algorithms and task scheduling. These behave like numpy arrays, but break a massive job into tasks that are then executed by a scheduler. The default scheduler uses threading but you can also use multiprocessing or distributed or even serial processing (mainly for debugging). You can tell the dask array how to break the data into chunks for processing.

```
import dask.array as da
f = h5py.File('myfile.hdf5')
x = da.from_array(f['/big-data'], chunks=(1000, 1000))
x - x.mean(axis=1).compute()
```

For more details on Dask Array check  
<https://dask.pydata.org/en/latest/array.html>

### 17.13.5 Dask DataFrame

A Dask DataFrame is a large parallel dataframe composed of many smaller Pandas dataframes, split along the index. These pandas

dataframes may live on disk for larger-than-memory computing on a single machine, or on many different machines in a cluster. Dask.dataframe implements a commonly used subset of the Pandas interface including elementwise operations, reductions, grouping operations, joins, timeseries algorithms, and more. It copies the Pandas interface for these operations exactly and so should be very familiar to Pandas users. Because Dask.dataframe operations merely coordinate Pandas operations they usually exhibit similar performance characteristics as are found in Pandas. To run the following code, save 'student.csv' file in your machine.

```
import pandas as pd
df = pd.read_csv('student.csv')
d = df.groupby(df.HID).Serial_No.mean()
print(d)
```

```
ID
101 1
102 2
104 3
105 4
106 5
107 6
109 7
111 8
201 9
202 10
Name: Serial_No, dtype: int64
```

```
import dask.dataframe as dd
df = dd.read_csv('student.csv')
dt = df.groupby(df.HID).Serial_No.mean().compute()
print(dt)
```

```
ID
101 1.0
102 2.0
104 3.0
105 4.0
106 5.0
107 6.0
109 7.0
111 8.0
201 9.0
202 10.0
Name: Serial_No, dtype: float64
```

For more details on Dask DataFrame check  
<https://dask.pydata.org/en/latest/dataframe.html>

## 17.13.6 Dask DataFrame Storage

Efficient storage can dramatically improve performance, particularly when operating repeatedly from disk.

Decompressing text and parsing CSV files is expensive. One of the most effective strategies with medium data is to use a binary storage format like HDF5.

```
be sure to shut down other kernels running distributed clients
from dask.distributed import Client
client = Client()
```

Create data if we don't have any

```
from prep import accounts_csvs
accounts_csvs(3, 1000000, 500)
```

First we read our csv data as before.

CSV and other text-based file formats are the most common storage for data from many sources, because they require minimal pre-processing, can be written line-by-line and are human-readable. Since Pandas' `read_csv` is well-optimized, CSVs are a reasonable input, but far from optimized, since reading required extensive text parsing.

```
import os
filename = os.path.join('data', 'accounts.*.csv')
filename

import dask.dataframe as dd
df_csv = dd.read_csv(filename)
df_csv.head()
```

HDF5 and netCDF are binary array formats very commonly used in the scientific realm.

Pandas contains a specialized HDF5 format, `HDFStore`. The `dd.DataFrame.to_hdf` method works exactly like the `pd.DataFrame.to_hdf` method.

```
target = os.path.join('data', 'accounts.h5')
target
```

```
%time df_csv.to_hdf(target, '/data')

df_hdf = dd.read_hdf(target, '/data')
df_hdf.head()
```

For more information of Dask DataFrame Storage, click <http://dask.pydata.org/en/latest/dataframe-create.html>

## 17.13.7 Links

- <https://dask.pydata.org/en/latest/>
- <http://matthewrocklin.com/blog/work/2017/10/16/streaming-dataframes-1>
- [http://people.duke.edu/~ccc14/sta-663-2017/18A\\_Dask.html](http://people.duke.edu/~ccc14/sta-663-2017/18A_Dask.html)
- <https://www.kdnuggets.com/2016/09/introducing-dask-parallel-programming.html>
- <https://pypi.python.org/pypi/dask/>
- <https://www.hdfgroup.org/2015/03/hdf5-as-a-zero-configuration-ad-hoc-scientific-database-for-python/>
- <https://github.com/dask/dask-tutorial>

## 17.14 DASK - RANDOM FOREST FEATURE DETECTION



### 17.14.1 Setup

First we need our tools. pandas gives us the DataFrame, very similar to R's DataFrames. The DataFrame is a structure that allows us to work with our data more easily. It has nice features for slicing and transformation of data, and easy ways to do basic statistics.

numpy has some very handy functions that work on DataFrames.

### 17.14.2 Dataset

We are using a dataset about the wine quality dataset, archived at UCI's Machine Learning Repository (<http://archive.ics.uci.edu/ml/index.php>).

```
import pandas as pd
```

```
import numpy as np
```

Now we'll load our data. pandas makes it easy!

```
red wine quality data, packed in a DataFrame
red_df = pd.read_csv('winequality-red.csv', sep=';', header=0, index_col=False)

white wine quality data, packed in a DataFrame
white_df = pd.read_csv('winequality-white.csv', sep=';', header=0, index_col=False)

rose? other fruit wines? plum wine? :(
```

Like in R, there is a `.describe()` method that gives basic statistics for every column in the dataset.

```
for red wines
red_df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar
<b>count</b>	1599.000000	1599.000000	1599.000000	1599.000000
<b>mean</b>	8.319637	0.527821	0.270976	2.538806
<b>std</b>	1.741096	0.179060	0.194801	1.409928
<b>min</b>	4.600000	0.120000	0.000000	0.900000
<b>25%</b>	7.100000	0.390000	0.090000	1.900000
<b>50%</b>	7.900000	0.520000	0.260000	2.200000
<b>75%</b>	9.200000	0.640000	0.420000	2.600000
<b>max</b>	15.900000	1.580000	1.000000	15.500000

```
for white wines
white_df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar
<b>count</b>	4898.000000	4898.000000	4898.000000	4898.000000
<b>mean</b>	6.854788	0.278241	0.334192	6.391415
<b>std</b>	0.843868	0.100795	0.121020	5.072058

<b>min</b>	3.800000	0.080000	0.000000	0.600000
<b>25%</b>	6.300000	0.210000	0.270000	1.700000
<b>50%</b>	6.800000	0.260000	0.320000	5.200000
<b>75%</b>	7.300000	0.320000	0.390000	9.900000
<b>max</b>	14.200000	1.100000	1.660000	65.800000

Sometimes it is easier to understand the data visually. A histogram of the white wine quality data citric acid samples is shown below. You can of course visualize other columns' data or other datasets. Just replace the DataFrame and column name below. See [Figure 169](#)

```
import matplotlib.pyplot as plt

def extract_col(df,col_name):
 return list(df[col_name])

col = extract_col(white_df,'citric acid') # can replace with another dataframe or column
plt.hist(col)

#TODO: add axes and such to set a good example

plt.show()
```

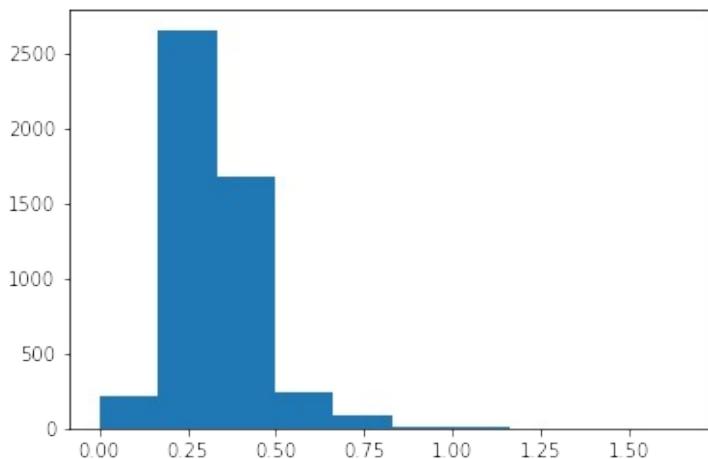


Figure 169: Histogram

### 17.14.3 Detecting Features

Let us try out a some elementary machine learning models. These

models are not always for prediction. They are also useful to find what features are most predictive of a variable of interest. Depending on the classifier you use, you may need to transform the data pertaining to that variable.

### 17.14.3.1 Data Preparation

Let us assume we want to study what features are most correlated with pH. pH of course is real-valued, and continuous. The classifiers we want to use usually need labeled or integer data. Hence, we will transform the pH data, assigning wines with pH higher than average as `hi` (more basic or alkaline) and wines with pH lower than average as `lo` (more acidic).

```
refresh to make Jupyter happy
red_df = pd.read_csv('winequality-red.csv', sep=';', header=0, index_col=False)
white_df = pd.read_csv('winequality-white.csv', sep=';', header=0, index_col=False)

#TODO: data cleansing functions here, e.g. replacement of NaN

if the variable you want to predict is continuous, you can map ranges of values
to integer/binary/string labels

for example, map the pH data to 'hi' and 'lo' if a pH value is more than or
less than the mean pH, respectively
M = np.mean(list(red_df['pH'])) # expect inelegant code in these mappings
Lf = lambda p: int(p < M)*'lo' + int(p >= M)*'hi' # some C-style hackery

create the new classifiable variable
red_df['pH-hi-lo'] = map(Lf, list(red_df['pH']))

and remove the predecessor
del red_df['pH']
```

Now we specify which dataset and variable you want to predict by assigning values to `SELECTED_DF` and `TARGET_VAR`, respectively.

We like to keep a parameter file where we specify data sources and such. This lets me create generic analytics code that is easy to reuse.

After we have specified what dataset we want to study, we split the training and test datasets. We then scale (normalize) the data, which makes most classifiers run better.

```

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

make selections here without digging in code
SELECTED_DF = red_df # selected dataset
TARGET_VAR = 'pH-hi-lo' # the predicted variable

generate nameless data structures
df = SELECTED_DF
target = np.array(df[TARGET_VAR]).ravel()
del df[TARGET_VAR] # no cheating

#TODO: data cleansing function calls here

split datasets for training and testing
X_train, X_test, y_train, y_test = train_test_split(df, target, test_size=0.2)

set up the scaler
scaler = StandardScaler()
scaler.fit(X_train)

apply the scaler
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

```

Now we pick a classifier. As you can see, there are many to try out, and even more in scikit-learn's documentation and many examples and tutorials. Random Forests are data science workhorses. They are the go-to method for most data scientists. Be careful relying on them though—they tend to overfit. We try to avoid overfitting by separating the training and test datasets.

## 17.14.4 Random Forest

```

pick a classifier

from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, ExtraTreeClassifier, E
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

clf = RandomForestClassifier()

```

Now we will test it out with the default parameters.

Note that this code is boilerplate. You can use it interchangeably for most scikit-learn models.

```
test it out
```

```

model = clf.fit(X_train,y_train)
pred = clf.predict(X_test)
conf_matrix = metrics.confusion_matrix(y_test,pred)

var_score = clf.score(X_test,y_test)

the results
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]

```

Now output the results. For Random Forests, we get a feature ranking. Relative importances usually exponentially decay. The first few highly-ranked features are usually the most important.

```

for the sake of clarity
num_features = X_train.shape[1]
features = map(lambda x: df.columns[x],indices)
feature_importances = map(lambda x: importances[x],indices)

print 'Feature ranking:\n'

for i in range(num_features):
 feature_name = features[i]
 feature_importance = feature_importances[i]
 print '%s%f' % (feature_name.ljust(30), feature_importance)

```

Feature ranking:

fixed acidity 0.269778 citric acid 0.171337 density 0.089660 volatile acidity 0.088965 chlorides 0.082945 alcohol 0.080437 total sulfur dioxide 0.067832 sulphates 0.047786 free sulfur dioxide 0.042727 residual sugar 0.037459 quality 0.021075

Sometimes it's easier to visualize. We'll use a bar chart. See [Figure 170](#)

```

plt.clf()
plt.bar(range(num_features),feature_importances)
plt.xticks(range(num_features),features,rotation=90)
plt.ylabel('relative importance (a.u.)')
plt.title('Relative importances of most predictive features')
plt.show()

```

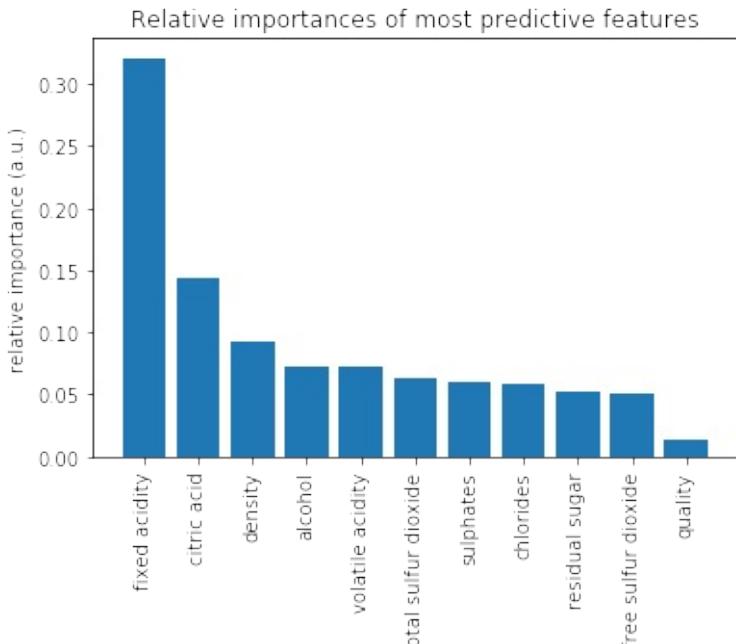


Figure 170: Result

```
import dask.dataframe as dd

red_df = dd.read_csv('winequality-red.csv', sep=';', header=0)
white_df = dd.read_csv('winequality-white.csv', sep=';', header=0)
```

## 17.14.5 Acknowledgement

This notebook was developed by Juliette Zerick and Gregor von Laszewski

## 17.15 FINGERPRINT MATCHING



Python is a flexible and popular language for running data analysis pipelines. In this section we will implement a solution for a fingerprint matching.

### 17.15.1 Overview

Fingerprint recognition refers to the automated method for verifying a match between two fingerprints and that is used to identify individuals and verify their identity. Fingerprints ([Figure 171](#)) are the most widely used form of biometric used to identify individuals.

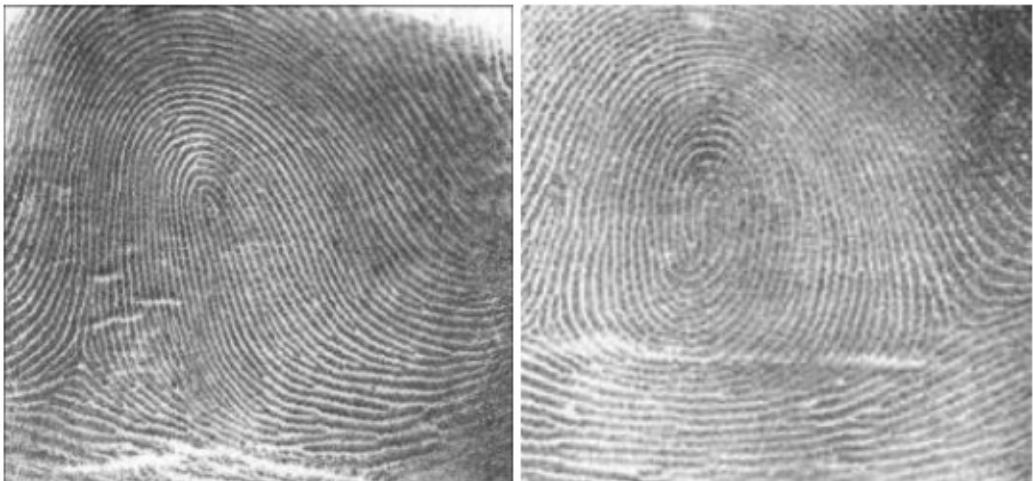


Figure 171: Fingerprints

The automated fingerprint matching generally required the detection of different fingerprint features (aggregate characteristics of ridges, and minutia points) and then the use of fingerprint matching algorithm, which can do both one-to-one and one-to-many matching operations. Based on the number of matches a proximity score (distance or similarity) can be calculated.

We use the following NIST dataset for the study:

Special Database 14 - NIST Mated Fingerprint Card Pairs 2.  
([http://www.nist.gov/itl/iad/ig/special\\_dbases.cfm](http://www.nist.gov/itl/iad/ig/special_dbases.cfm))

## 17.15.2 Objectives

Match the fingerprint images from a probe set to a gallery set and report the match scores.

## 17.15.3 Prerequisites

For this work we will use the following algorithms:

- MINDTCT: The NIST minutiae detector, which automatically locates and records ridge ending and bifurcations in a fingerprint image. (<http://www.nist.gov/itl/iad/ig/nbis.cfm>)
- BOZORTH3: A NIST fingerprint matching algorithm, which is a

minutiae based fingerprint-matching algorithm. It can do both one-to-one and one-to-many matching operations. (<http://www.nist.gov/itl/iad/ig/nbis.cfm>)

In order to follow along, you must have the NBIS tools which provide `mindtct` and `bozorth3` installed. If you are on Ubuntu 16.04 Xenial, the following steps will accomplish this:

```
$ sudo apt-get update -qq
$ sudo apt-get install -y build-essential cmake unzip
$ wget "http://nigos.nist.gov:8080/nist/nbis/nbis_v5_0_0.zip"
$ unzip -d nbis nbis_v5_0_0.zip
$ cd nbis/Rel_5.0.0
$./setup.sh /usr/local --without-X11
$ sudo make
```

## 17.15.4 Implementation

1. Fetch the fingerprint images from the web
2. Call out to external programs to prepare and compute the match scores
3. Store the results in a database
4. Generate a plot to identify likely matches.

```
from __future__ import print_function

import urllib
import zipfile
import hashlib
```

We'll be interacting with the operating system and manipulating files and their pathnames.

```
import os.path
import os
import sys
import shutil
import tempfile
```

## Some general useful utilities

```
import itertools
import functools
import types
from pprint import pprint
```

Using the `attrs` library provides some nice shortcuts to defining objects

```
import attr
import sys
```

We'll be randomly dividing the entire dataset, based on user input, into the probe and gallery sets

```
import random
```

We'll need to call out to the NBIS software. We'll also be using multiple processes to take advantage of all the cores on our machine

```
import subprocess
import multiprocessing
```

As for plotting, we'll use `matplotlib`, though there are many alternatives.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Finally, we'll write the results to a database.

```
import sqlite3
```

## 17.15.5 Utility functions

Next, we'll define some utility functions:

```
def take(n, iterable):
 """Returns a generator of the first **n** elements of an iterable"
 return itertools.islice(iterable, n)

def zipWith(function, *iterables):
 """Zip a set of **iterables** together and apply **function** to each tuple"
 for group in itertools.izip(*iterables):
 yield function(*group)

def uncurry(function):
 """Transforms an N-arry **function** so that it accepts a single parameter of an N-tuple"
 @functools.wraps(function)
 def wrapper(args):
 return function(*args)
 return wrapper
```

```

def fetch_url(url, sha256, prefix='.', checksum_blocksize=2**20, dryRun=False):
 """Download a url.

 :param url: the url to the file on the web
 :param sha256: the SHA-256 checksum. Used to determine if the file was previously downloaded.
 :param prefix: directory to save the file
 :param checksum_blocksize: blocksize to used when computing the checksum
 :param dryRun: boolean indicating that calling this function should do nothing
 :returns: the local path to the downloaded file
 :rtype:

 """

 if not os.path.exists(prefix):
 os.makedirs(prefix)

 local = os.path.join(prefix, os.path.basename(url))

 if dryRun: return local

 if os.path.exists(local):
 print ('Verifying checksum')
 chk = hashlib.sha256()
 with open(local, 'rb') as fd:
 while True:
 bits = fd.read(checksum_blocksize)
 if not bits: break
 chk.update(bits)
 if sha256 == chk.hexdigest():
 return local

 print ('Downloading', url)

 def report(sofar, blocksize, totalsize):
 msg = '{}%\r'.format(100 * sofar * blocksize / totalsize, 100)
 sys.stderr.write(msg)

 urllib.urlretrieve(url, local, report)

 return local

```

## 17.15.6 Dataset

We'll now define some global parameters

First, the fingerprint dataset

```

DATASET_URL = 'https://s3.amazonaws.com/nist-srd/SD4/NISTSpecialDatabase4GrayScaleImagesofFingerprints'
DATASET_SHA256 = '4db6a8f3f9dc14c504180cbf67cdf35167a109280f121c901be37a80ac13c449'

```

We'll define how to download the dataset. This function is general enough that it could be used to retrieve most files, but we'll default it to use the values from above.

```
def prepare_dataset(url=None, sha256=None, prefix='.', skip=False):
 url = url or DATASET_URL
 sha256 = sha256 or DATASET_SHA256
 local = fetch_url(url, sha256=sha256, prefix=prefix, dryRun=skip)

 if not skip:
 print ('Extracting', local, 'to', prefix)
 with zipfile.ZipFile(local, 'r') as zip:
 zip.extractall(prefix)

 name, _ = os.path.splitext(local)
 return name

def locate_paths(path_md5list, prefix):
 with open(path_md5list) as fd:
 for line in itertools.imap(str.strip, fd):
 parts = line.split()
 if not len(parts) == 2: continue
 md5sum, path = parts
 checksum = Checksum(value=md5sum, kind='md5')
 filepath = os.path.join(prefix, path)
 yield Path(checksum=checksum, filepath=filepath)

def locate_images(paths):
 def predicate(path):
 _, ext = os.path.splitext(path.filepath)
 return ext in ['.png']

 for path in itertools.ifilter(predicate, paths):
 yield image(id=path.checksum.value, path=path)
```

## 17.15.7 Data Model

We'll define some classes so we have a nice API for working with the dataflow. We set `slots=True` so that the resulting objects will be more space-efficient.

### 17.15.7.1 Utilities

#### 17.15.7.1.1 Checksum

The checksum consists of the actual hash value (`value`) as well as a string representing the hashing algorithm. The validator enforces that the algorithm can only be one of the listed acceptable methods

```
@attr.s(slots=True)
class Checksum(object):
 value = attr.ib()
 kind = attr.ib(validation=lambda o, a, v: v in 'md5 sha1 sha224 sha256 sha384 sha512'.split())
 |
```

## 17.15.7.1.2 Path

`Paths` refer to an image's filepath and associated `checksum`. We get the checksum "for free" since the MD5 hash is provided for each image in the dataset.

```
@attr.s(slots=True)
class Path(object):
 checksum = attr.ib()
 filepath = attr.ib()
```

### **17.15.7.1.3 Image**

The start of the data pipeline is the image. An `image` has an `id` (the md5 hash) and the path to the image.

```
@attr.s(slots=True)
class image(object):
 id = attr.ib()
 path = attr.ib()
```

## 17.15.7.2 Mindtct

The next step in the pipeline is to apply the `mindtct` program from NBIS. A `mindtct` object therefore represents the results of applying `mindtct` on an `image`. The `xyt` output is needed for the next step, and the `image` attribute represents the image id.

```
@attr.s(slots=True)
class mindtct(object):
 image = attr.ib()
 xyt = attr.ib()

 def pretty(self):
 d = dict(id=self.image.id, path=self.image.path)
```

```
 return pprint(d)
```

We need a way to construct a `mindtct` object from an `image` object. A straightforward way of doing this would be to have a `from_image` `@staticmethod` or `@classmethod`, but that doesn't work well with `multiprocessing` as top-level functions work best as they need to be serialized.

```
def mindtct_from_image(image):
 imgpath = os.path.abspath(image.pathfilepath)
 tempdir = tempfile.mkdtemp()
 oroot = os.path.join(tempdir, 'result')

 cmd = ['mindtct', imgpath, oroot]

 try:
 subprocess.check_call(cmd)

 with open(rooot + '.xyt') as fd:
 xyt = fd.read()

 result = mindtct(image=image.id, xyt=xyt)
 return result

 finally:
 shutil.rmtree(tempdir)
```

### 17.15.7.3 Bozorth3

The final step in the pipeline is running the `bozorth3` from NBIS. The `bozorth3` class represents the match being done: tracking the ids of the probe and gallery images as well as the match score.

Since we'll be writing these instances out to a database, we provide some static methods for SQL statements. While there are many Object-Relational-Model (ORM) libraries available for Python, this approach keeps the current implementation simple.

```
@attr.s(slots=True)
class bozorth3(object):
 probe = attr.ib()
 gallery = attr.ib()
 score = attr.ib()

 @staticmethod
 def sql_stmt_create_table():
 return 'CREATE TABLE IF NOT EXISTS bozorth3' \
 + '(probe TEXT, gallery TEXT, score NUMERIC)'
```

```

@staticmethod
def sql_prepared_stmt_insert():
 return 'INSERT INTO bozorth3 VALUES (?, ?, ?)'

def sql_prepared_stmt_insert_values(self):
 return self.probe, self.gallery, self.score

```

In order to work well with `multiprocessing`, we define a class representing the input parameters to `bozorth3` and a helper function to run `bozorth3`. This way the pipeline definition can be kept simple to a `map` to create the input and then a `map` to run the program.

As NBIS `bozorth3` can be called to compare one-to-one or one-to-many, we'll also dynamically choose between these approaches depending on if the gallery attribute is a list or a single object.

```

@attr.s(slots=True)
class bozorth3_input(object):
 probe = attr.ib()
 gallery = attr.ib()

 def run(self):
 if isinstance(self.gallery, mindtct):
 return bozorth3_from_one_to_one(self.probe, self.gallery)
 elif isinstance(self.gallery, types.ListType):
 return bozorth3_from_one_to_many(self.probe, self.gallery)
 else:
 raise ValueError('Unhandled type for gallery: {}'.format(type(gallery)))

```

The next is the top-level function to running `bozorth3`. It accepts an instance of `bozorth3_input`. This is implemented as a simple top-level wrapper so that it can be easily passed to the `multiprocessing` library.

```

def run_bozorth3(input):
 return input.run()

```

### 17.15.7.3.1 Running Bozorth3

There are two cases to handle: 1. One-to-one probe to gallery sets 1. One-to-many probe to gallery sets

Both approaches are implemented below. The implementations follow the same pattern: 1. Create a temporary directory within to work 1. Write the probe and gallery images to files in the temporary directory 1. Call the `bozorth3` executable 1. The match score

is written to `stdout` which is captured and then parsed. 1. Return a `bozorth3` instance for each match 1. Make sure to clean up the temporary directory

#### 17.15.7.3.1.1 One-to-one

```
def bozorth3_from_one_to_one(probe, gallery):
 tempdir = tempfile.mkdtemp()
 probeFile = os.path.join(tempdir, 'probe.xyt')
 galleryFile = os.path.join(tempdir, 'gallery.xyt')

 with open(probeFile, 'wb') as fd: fd.write(probe.xyt)
 with open(galleryFile, 'wb') as fd: fd.write(gallery.xyt)

 cmd = ['bozorth3', probeFile, galleryFile]

 try:
 result = subprocess.check_output(cmd)
 score = int(result.strip())
 return bozorth3(probe=probe.image, gallery=gallery.image, score=score)
 finally:
 shutil.rmtree(tempdir)
```

#### 17.15.7.3.1.2 One-to-many

```
def bozorth3_from_one_to_many(probe, galleryset):
 tempdir = tempfile.mkdtemp()
 probeFile = os.path.join(tempdir, 'probe.xyt')
 galleryFiles = [os.path.join(tempdir, 'gallery%d.xyt' % i)
 for i,_ in enumerate(galleryset)]

 with open(probeFile, 'wb') as fd: fd.write(probe.xyt)
 for galleryFile, gallery in itertools.izip(galleryFiles, galleryset):
 with open(galleryFile, 'wb') as fd: fd.write(gallery.xyt)

 cmd = ['bozorth3', '-p', probeFile] + galleryFiles

 try:
 result = subprocess.check_output(cmd).strip()
 scores = map(int, result.split('\n'))
 return [bozorth3(probe=probe.image, gallery=gallery.image, score=score)
 for score, gallery in zip(scores, galleryset)]
 finally:
 shutil.rmtree(tempdir)
```

## 17.16 PLOTTING

---

For plotting we'll operate only on the database. We'll select a small number of probe images and plot the score between them and the

rest of the gallery images.

The `mk_short_labels` helper function will be defined below.

```
def plot(dbfile, nprobes=10):
 conn = sqlite3.connect(dbfile)
 results = pd.read_sql(
 "SELECT DISTINCT probe FROM bozorth3 ORDER BY score LIMIT '%s'" % nprobes,
 con=conn
)
 shortlabels = mk_short_labels(results.probe)
 plt.figure()

 for i, probe in results.probe.iteritems():
 stmt = 'SELECT gallery, score FROM bozorth3 WHERE probe = ? ORDER BY gallery DESC'
 matches = pd.read_sql(stmt, params=(probe,), con=conn)
 xs = np.arange(len(matches), dtype=np.int)
 plt.plot(xs, matches.score, label='probe %s' % shortlabels[i])

 plt.ylabel('Score')
 plt.xlabel('Gallery')
 plt.legend(bbox_to_anchor=(0, 0, 1, -0.2))
 plt.show()
```

The image ids are long hash strings. In order to minimize the amount of space on the figure the labels occupy, we provide a helper function to create a short label that still uniquely identifies each probe image in the selected sample

```
def mk_short_labels(series, start=7):
 for size in xrange(start, len(series[0])):
 if len(series) == len(set(map(lambda s: s[:size], series))):
 break
 return map(lambda s: s[:size], series)
```

## 17.17 PUTTING IT ALL TOGETHER

---

First, set up a temporary directory in which to work:

```
pool = multiprocessing.Pool()
prefix = '/tmp/fingerprint_example/'
if not os.path.exists(prefix):
 os.makedirs(prefix)
```

Next we download and extract the fingerprint images from NIST:

```
%%time
dataprefix = prepare_dataset(prefix=prefix)
```

```
Verifying checksum Extracting
/tmp/fingerprint_example/NISTSpecialDatabase4GrayScaleImagesofFIGS.zip
to /tmp/fingerprint_example/ CPU times: user 3.34 s, sys: 645 ms,
total: 3.99 s Wall time: 4.01 s
```

Next we'll configure the location of of the MD5 checksum file that comes with the download

```
md5listpath = os.path.join(prefix, 'NISTSpecialDatabase4GrayScaleImagesofFIGS/sd04/sd04_md5list.txt')
```

Load the images from the downloaded files to start the analysis pipeline

```
%%time
print('Loading images')
paths = locate_paths(md5listpath, dataprefix)
images = locate_images(paths)
mindtcts = pool.map(mindtct_from_image, images)
print('Done')
```

```
Loading images Done CPU times: user 187 ms, sys: 17 ms, total: 204 ms
Wall time: 1min 21s
```

We can examine one of the loaded image. Note that `image` is refers to the MD5 checksum that came with the image and the `xyt` attribute represents the raw image data.

```
print(mindtcts[0].image)
print(mindtcts[0].xyt[:50])
```

```
98b15d56330cb17f1982ae79348f711d 14 146 214 6 25 238 22 37 25 51 180 20
30 332 214
```

For example purposes we'll only a use a small percentage of the database, randomly selected, for pur probe and gallery datasets.

```
perc_probe = 0.001
perc_gallery = 0.1
```

```
%%time
print('Generating samples')
probes = random.sample(mindtcts, int(perc_probe * len(mindtcts)))
gallery = random.sample(mindtcts, int(perc_gallery * len(mindtcts)))
print('|Probes| =', len(probes))
print('|Gallery| =', len(gallery))
```

```
Generating samples = 4 = 400 CPU times: user 2 ms, sys: 0 ns, total: 2
ms Wall time: 993 µs
```

We can now compute the matching scores between the probe and gallery sets. This will use all cores available on this workstation.

```
%time
print('Matching')
input = [bozorth3_input(probe=probe, gallery=gallery)
 for probe in probes]
bozorth3s = pool.map(run_bozorth3, input)

Matching CPU times: user 19 ms, sys: 1 ms, total: 20 ms Wall time: 1.07
s
```

bozorth3s is now a list of lists of bozorth3 instances.

```
print('|Probes| =', len(bozorth3s))
print('|Gallery| =', len(bozorth3s[0]))
print('Result:', bozorth3s[0][0])

= 4 = 400 Result: bozorth3(probe='caf9143b268701416fbed6a9eb2eb4cf',
gallery='22fa0f24998eaea39dea152e4a73f267', score=4)
```

Now add the results to the database

```
dbfile = os.path.join(prefix, 'scores.db')
conn = sqlite3.connect(dbfile)
cursor = conn.cursor()
cursor.execute(bozorth3.sql_stmt_create_table())

<sqlite3.Cursor at 0x7f8a2f677490>

%%time
for group in bozorth3s:
 vals = map(bozorth3.sql_prepared_stmt_insert_values, group)
 cursor.executemany(bozorth3.sql_prepared_stmt_insert(), vals)
 conn.commit()
 print('Inserted results for probe', group[0].probe)

Inserted results for probe caf9143b268701416fbed6a9eb2eb4cf Inserted
results for probe 55ac57f711eba081b9302eab74dea88e Inserted results for
probe 4ed2d53db3b5ab7d6b216ea0314beb4f Inserted results for probe
20f68849ee2dad02b8fb33ecd3ece507 CPU times: user 2 ms, sys: 3 ms, total:
5 ms Wall time: 3.57 ms
```

We now plot the results. [Figure 172](#)

```
plot(dbfile, nprobes=len(probes))
```

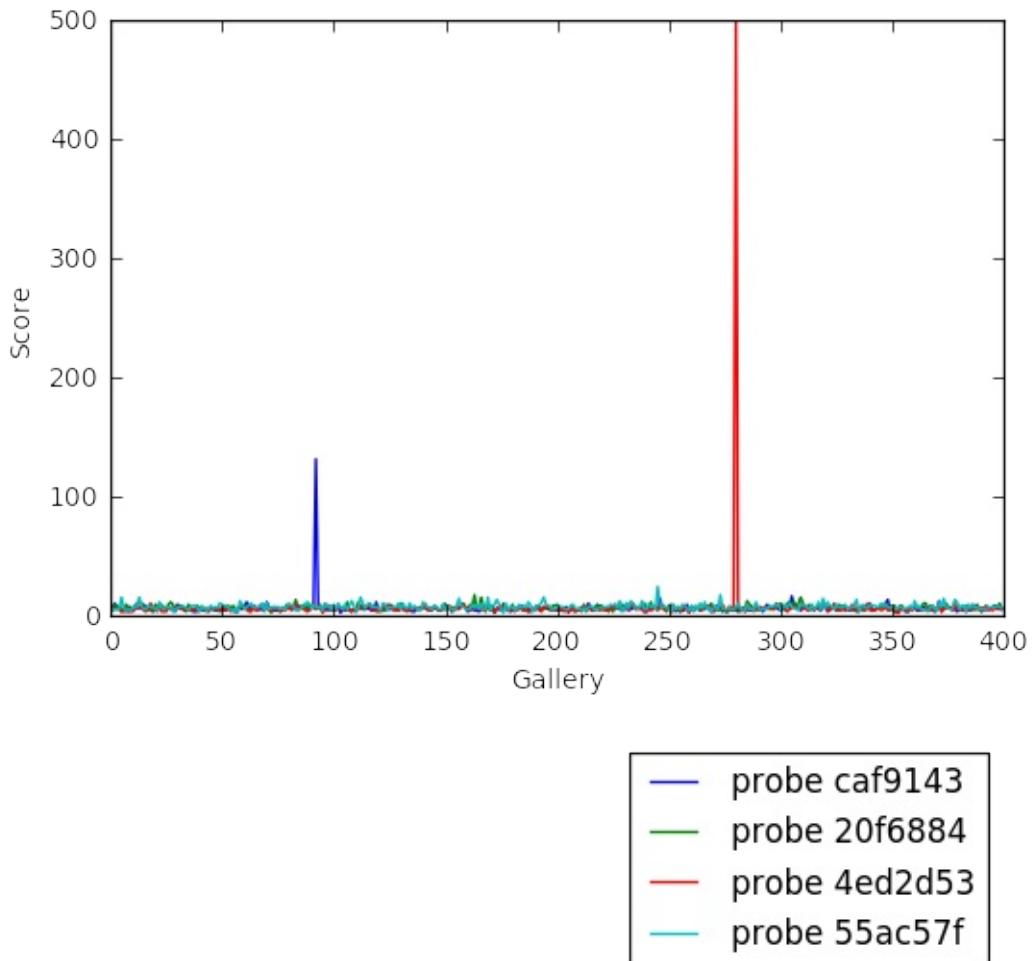


Figure 172: Result

```
cursor.close()
```

## **17.18 NIST PEDESTRIAN AND FACE DETECTION**

Pedestrian and Face Detection uses OpenCV to identify people standing in a picture or a video and NIST use case in this document is built with Apache Spark and Mesos clusters on multiple compute nodes.

The example in this tutorial deploys software packages on OpenStack using Ansible with its roles. See [Figure 173](#), [Figure 174](#), [Figure 175](#), [Figure 176](#)



Figure 173: Original

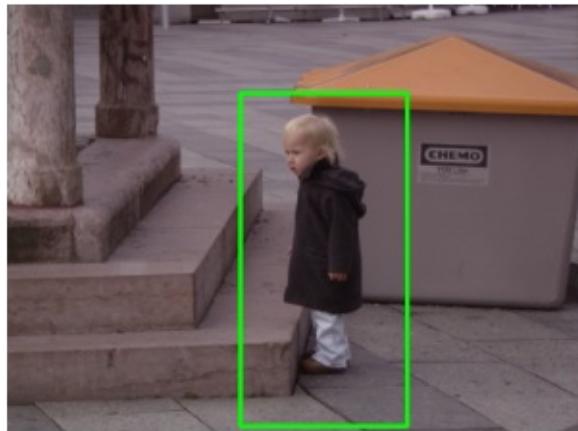


Figure 174: Pedestrian Detected



Figure 175: Original



Figure 176: Pedestrian and Face/eyes Detected

### 17.18.0.1 Introduction

Human (pedestrian) detection and face detection have been studied during the last several years and models for them have improved along with Histograms of Oriented Gradients (HOG) for Human Detection [1]. OpenCV is a Computer Vision library including the SVM classifier and the HOG object detector for pedestrian detection and INRIA Person Dataset [2] is one of popular samples for both training and testing purposes. In this document, we deploy Apache Spark on Mesos clusters to train and apply detection models from OpenCV using Python API.

#### 17.18.0.1.1 INRIA Person Dataset

This dataset contains positive and negative images for training and test purposes with annotation files for upright persons in each image. 288 positive test images, 453 negative test images, 614 positive training images and 1218 negative training images are included along with normalized 64x128 pixel formats. 970MB dataset is available to download [3].

#### 17.18.0.1.2 HOG with SVM model

Histogram of Oriented Gradient (HOG) and Support Vector Machine

(SVM) are used as object detectors and classifiers and built-in python libraries from OpenCV provide these models for human detection.

#### **17.18.0.1.3 Ansible Automation Tool**

Ansible is a python tool to install/configure/manage software on multiple machines with JSON files where system descriptions are defined. There are reasons why we use Ansible:

- Expandable: Leverages Python (default) but modules can be written in any language
- Agentless: no setup required on managed node
- Security: Allows deployment from user space; uses ssh for authentication
- Flexibility: only requires ssh access to privileged user
- Transparency: YAML Based script files express the steps of installing and configuring software
- Modularity: Single Ansible Role (should) contain all required commands and variables to deploy software package independently
- Sharing and portability: roles are available from source (github, bitbucket, gitlab, etc) or the Ansible Galaxy portal

We use Ansible roles to install software packages for Human and Face Detection which requires to run OpenCV Python libraries on Apache Mesos with a cluster configuration. Dataset is also downloaded from the web using an ansible role.

#### **17.18.0.2 Deployment by Ansible**

Ansible is to deploy applications and build clusters for batch-processing large datasets towards target machines e.g. VM instances

on OpenStack and we use ansible roles with include directive to organize layers of big data software stacks (BDSS). Ansible provides abstractions by Playbook Roles and reusability by Include statements. We define X application in X Ansible Role, for example, and use include statements to combine with other applications e.g. Y or Z. The layers exist in sub directories (see below) to add modularity to your Ansible deployment. For example, there are five roles used in this example that are Apache Mesos in a scheduler layer, Apache Spark in a processing layer, a OpenCV library in an application layer, INRIA Person Dataset in a dataset layer and a python script for human and face detection in an analytics layer. If you have an additional software package to add, you can simply add a new role in a main ansible playbook with include directive. With this, your Ansible playbook maintains simple but flexible to add more roles without having a large single file which is getting difficult to read when it deploys more applications on multiple layers. The main Ansible playbook runs Ansible roles in order which look like:

```
```
include: sched/00-mesos.yml
include: proc/01-spark.yml
include: apps/02-opencv.yml
include: data/03-inria-dataset.yml
Include: anlys/04-human-face-detection.yml
```
```

Directory names e.g. sched, proc, data, or anlys indicate BDSS layers like: - sched: scheduler layer - proc: data processing layer - apps: application layer - data: dataset layer - anlys: analytics layer and two digits in the filename indicate an order of roles to be run.

### 17.18.0.3 Cloudmesh for Provisioning

It is assumed that virtual machines are created by cloudmesh, the cloud management software. For example on OpenStack,

```
cm cluster create -N=6
```

command starts a set of virtual machine instances. The number of machines and groups for clusters e.g. namenodes and datanodes are

defined in the Ansible inventory file, a list of target machines with groups, which will be generated once machines are ready to use by cloudmesh. Ansible roles install software and dataset on virtual clusters after that stage.

#### **17.18.0.4 Roles Explained for Installation**

Mesos role is installed first as a scheduler layer for masters and slaves where mesos-master runs on the masters group and mesos-slave runs on the slaves group. Apache Zookeeper is included in the mesos role therefore mesos slaves find an elected mesos leader for the coordination. Spark, as a data processing layer, provides two options for distributed job processing, batch job processing via a cluster mode and real-time processing via a client mode. The Mesos dispatcher runs on a masters group to accept a batch job submission and Spark interactive shell, which is the client mode, provides real-time processing on any node in the cluster. Either way, Spark is installed later to detect a master (leader) host for a job submission. Other roles for OpenCV, INRIA Person Dataset and Human and Face Detection Python applications are followed by.

The following software are expected in the stacks according to the [github](#):

- mesos cluster (master, worker)
- spark (with dispatcher for mesos cluster mode)
- openCV
- zookeeper
- INRIA Person Dataset
- Detection Analytics in Python
- [1] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE Computer Society

Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 1. IEEE,

2005. [pdf]

- [2] <http://pascal.inrialpes.fr/data/human/>
- [3] <ftp://ftp.inrialpes.fr/pub/lear/douze/data/INRIAPerson.tar>
- [4] <https://docs.python.org/2/library/configparser.html>

#### 17.18.0.4.1 Server groups for Masters/Slaves by Ansible inventory

We may separate compute nodes in two groups: masters and workers therefore Mesos masters and zookeeper quorums manage job requests and leaders and workers run actual tasks. Ansible needs group definitions in their inventory therefore software installation associated with a proper part can be completed.

Example of Ansible Inventory file (inventory.txt)

```
[masters]
10.0.5.67
10.0.5.68
10.0.5.69
[slaves]
10.0.5.70
10.0.5.71
10.0.5.72
```

#### 17.18.0.5 Instructions for Deployment

The following commands complete NIST Pedestrian and Face Detection deployment on OpenStack.

##### 17.18.0.5.1 Cloning Pedestrian Detection Repository from Github

Roles are included as submodules which require `--recursive` option to checkout them all.

```
$ git clone --recursive https://github.com/futuresystems/pedestrian-and-face-detection.git
```

Change the following variable with actual ip addresses:

```
sample_inventory="""
[masters]
10.0.5.67
10.0.5.68
10.0.5.69
[slaves]
10.0.5.70
10.0.5.71
10.0.5.72""""
```

Create a `inventory.txt` file with the variable in your local directory.

```
!printf "$sample_inventory" > inventory.txt
!cat inventory.txt
```

Add `ansible.cfg` file with options for ssh host key checking and login name.

```
ansible_config="""
[defaults]
host_key_checking=false
remote_user=ubuntu"""
!printf "$ansible_config" > ansible.cfg
!cat ansible.cfg
```

Check accessibility by ansible ping like:

```
!ansible -m ping -i inventory.txt all
```

Make sure that you have a correct ssh key in your account otherwise you may encounter 'FAILURE' in the ping test above.

### 17.18.0.5.2 Ansible Playbook

We use a main ansible playbook to deploy software packages for NIST Pedestrian and Face detection which includes: - mesos - spark - zookeeper - opencv - INRIA Person dataset - Python script for the detection

```
!cd pedestrian-and-face-detection/ && ansible-playbook -i/inventory.txt site.yml
```

The installation may take 30 minutes or an hour to complete.

### 17.18.0.6 OpenCV in Python

Before we run our code for this project, let's try OpenCV first to see how it works.

#### 17.18.0.6.1 Import cv2

Let's import opencv python module and we will use images from the online database image-net.org to test OpenCV image recognition. See [Figure 177](#), [Figure 178](#)

```
import cv2
```

Let's download a mailbox image with a red color to see if opencv identifies the shape with a color. The example file in this tutorial is:

```
$ curl http://farm4.static.flickr.com/3061/2739199963_ee78af76ef.jpg > mailbox.jpg
```

```
100 167k 100 167k 0 0 686k 0 -:-:- -:-:- -:-:- 684k
```

```
%matplotlib inline

from IPython.display import Image
mailbox_image = "mailbox.jpg"
Image(filename=mailbox_image)
```



Figure 177: Mailbox image

You can try other images. Check out the [image-net.org](#) for mailbox

images: <http://image-net.org/synset?wnid=n03710193>

#### 17.18.0.6.2 Image Detection

Just for a test, let's try to detect a red color shaped mailbox using opencv python functions.

There are key functions that we use:

- \* cvtColor: to convert a color space of an image
- \* inRange: to detect a mailbox based on the range of red color pixel values
- \* np.array: to define the range of red color using a Numpy library for better calculation
- \* findContours: to find a outline of the object
- \* bitwise\_and: to black-out the area of contours found

```
import numpy as np
import matplotlib.pyplot as plt

imread for loading an image
img = cv2.imread(mailbox_image)
cvtColor for color conversion
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

define range of red color in hsv
lower_red1 = np.array([0, 50, 50])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 50, 50])
upper_red2 = np.array([180, 255, 255])

threshold the hsv image to get only red colors
mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
mask = mask1 + mask2

find a red color mailbox from the image
im2, contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

bitwise_and to remove other areas in the image except the detected object
res = cv2.bitwise_and(img, img, mask = mask)

turn off - x, y axis bar
plt.axis("off")
text for the masked image
cv2.putText(res, "masked image", (20,300), cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255))
display
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
plt.show()
```



Figure 178: Masked image

The red color mailbox is left alone in the image which we wanted to find in this example by opencv functions. You can try other images with different colors to detect the different shape of objects using `findContours` and `inRange` from opencv.

For more information, see the useful links below.

- contours features:  
[http://docs.opencv.org/3.1.0/dd/d49/tutorial\\_py\\_contour\\_features.html](http://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html)
- contours:  
[http://docs.opencv.org/3.1.0/d4/d73/tutorial\\_py\\_contours\\_begin.html](http://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html)
- red color in hsv:  
<http://stackoverflow.com/questions/30331944/finding-red-color-using-python-opencv>
- inrange:  
[http://docs.opencv.org/master/da/d97/tutorial\\_threshold\\_inRange.html](http://docs.opencv.org/master/da/d97/tutorial_threshold_inRange.html)
- inrange:  
[http://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_imgproc/py\\_thresholding/py\\_thresholding.html](http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html)

[beta/doc/py\\_tutorials/py\\_imgproc/py\\_colorspaces/py\\_color.html](http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_colorspaces/py_color.html)

- numpy: [http://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_core/py\\_basic\\_ops/py\\_basic\\_ops.html](http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_basic_ops/py_basic_ops.html)

## 17.18.0.7 Human and Face Detection in OpenCV

### 17.18.0.7.1 INRIA Person Dataset

We use INRIA Person dataset to detect upright people and faces in images in this example. Let's download it first.

```
$ curl ftp://ftp.inrialpes.fr/pub/lear/douze/data/INRIAPerson.tar > INRIAPerson.tar
```

```
100 969M 100 969M 0 0 8480k 0 0:01:57 0:01:57 -:-:-
12.4M
```

```
$ tar xvf INRIAPerson.tar > logfile && tail logfile
```

### 17.18.0.7.2 Face Detection using Haar Cascades

This section is prepared based on the opencv-python tutorial:  
[http://docs.opencv.org/3.1.0/d7/d8b/tutorial\\_py\\_face\\_detection.htm](http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.htm)

There is a pre-trained classifier for face detection, download it from here:

```
$ curl https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade
```

```
100 908k 100 908k 0 0 2225k 0 -:-:- -:-:- -:-:- 2259k
```

This classifier XML file will be used to detect faces in images. If you like to create a new classifier, find out more information about training from here:  
[http://docs.opencv.org/3.1.0/dc/d88/tutorial\\_traincascade.html](http://docs.opencv.org/3.1.0/dc/d88/tutorial_traincascade.html)

### 17.18.0.7.3 Face Detection Python Code Snippet

Now, we detect faces from the first five images using the classifier.

See [Figure 179](#), [Figure 180](#), [Figure 181](#), [Figure 182](#), [Figure 183](#), [Figure 184](#), [Figure 185](#), [Figure 186](#), [Figure 187](#), [Figure 188](#), [Figure 189](#)

```
import the necessary packages
from __future__ import print_function
import numpy as np
import cv2
from os import listdir
from os.path import isfile, join
import matplotlib.pyplot as plt

mypath = "INRIAPerson/Test/pos/"
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

onlyfiles = [join(mypath, f) for f in listdir(mypath) if isfile(join(mypath, f))]

cnt = 0
for filename in onlyfiles:
 image = cv2.imread(filename)
 image_grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
 faces = face_cascade.detectMultiScale(image_grayscale, 1.3, 5)
 if len(faces) == 0:
 continue

 cnt_faces = 1
 for (x,y,w,h) in faces:
 cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
 cv2.putText(image, "face" + str(cnt_faces), (x,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1,
 plt.figure()
 plt.axis("off")
 plt.imshow(cv2.cvtColor(image[y:y+h, x:x+w], cv2.COLOR_BGR2RGB))
 cnt_faces += 1
 plt.figure()
 plt.axis("off")
 plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
 cnt = cnt + 1
 if cnt == 5:
 break
[◀] [▶]
```



Figure 179: Example



Figure 180: Example



Figure 181: Example

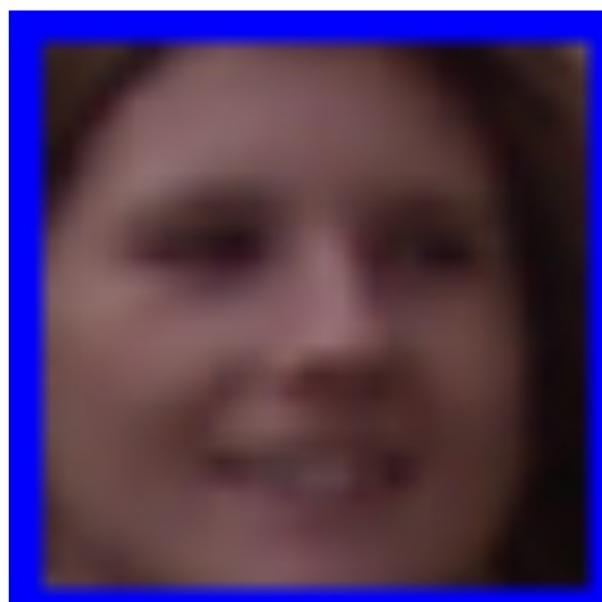


Figure 182: Example



Figure 183: Example



Figure 184: Example



Figure 185: Example



Figure 186: Example



Figure 187: Example



Figure 188: Example



Figure 189: Example

### 17.18.0.8 Pedestrian Detection using HOG Descriptor

We will use Histogram of Oriented Gradients (HOG) to detect a upright person from images. See [Figure 190](#), [Figure 191](#), [Figure 192](#), [Figure 193](#), [Figure 194](#), [Figure 195](#), [Figure 196](#), [Figure 197](#), [Figure 198](#), [Figure 199](#)

#### 17.18.0.8.1 Python Code Snippet

```
initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

cnt = 0
for filename in onlyfiles:
 img = cv2.imread(filename)
 orig = img.copy()
 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

 # detect people in the image
 (rects, weights) = hog.detectMultiScale(img, winStride=(8, 8),
 padding=(16, 16), scale=1.05)

 # draw the final bounding boxes
 for (x, y, w, h) in rects:
 cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

```
plt.figure()
plt.axis("off")
plt.imshow(cv2.cvtColor(orig, cv2.COLOR_BGR2RGB))
plt.figure()
plt.axis("off")
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
cnt = cnt + 1
if cnt == 5:
 break
```



Figure 190: Example

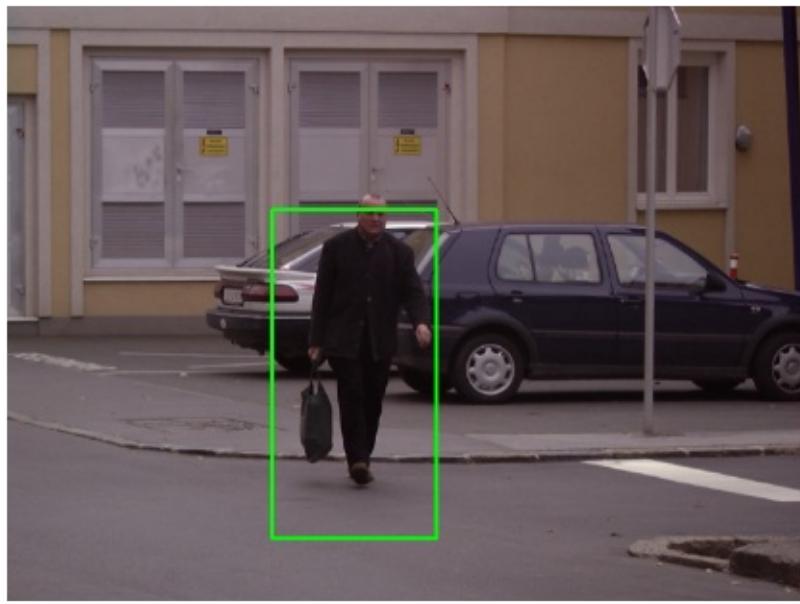


Figure 191: Example



Figure 192: Example

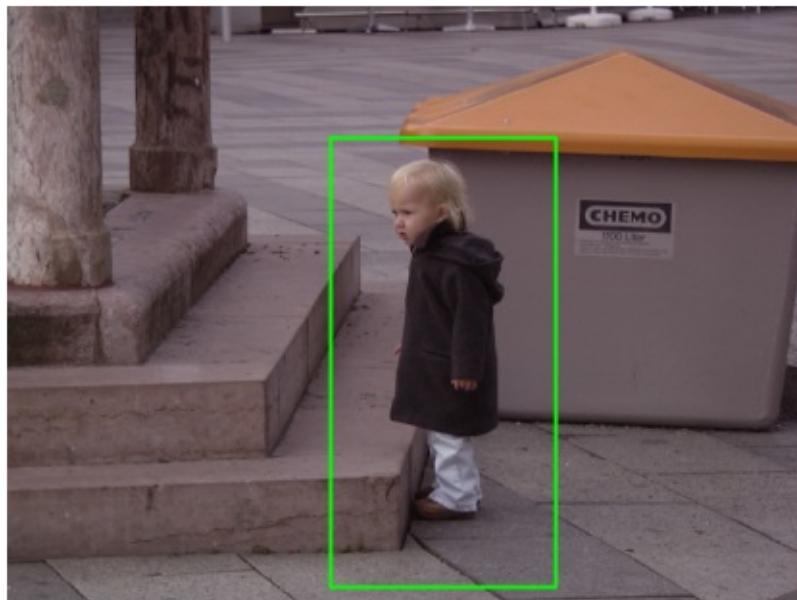


Figure 193: Example



Figure 194: Example

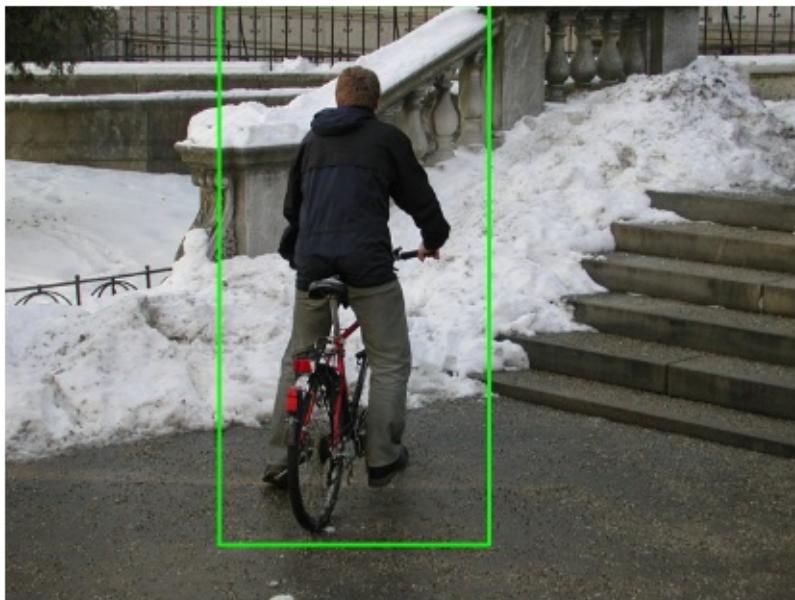


Figure 195: Example



Figure 196: Example



Figure 197: Example



Figure 198: Example

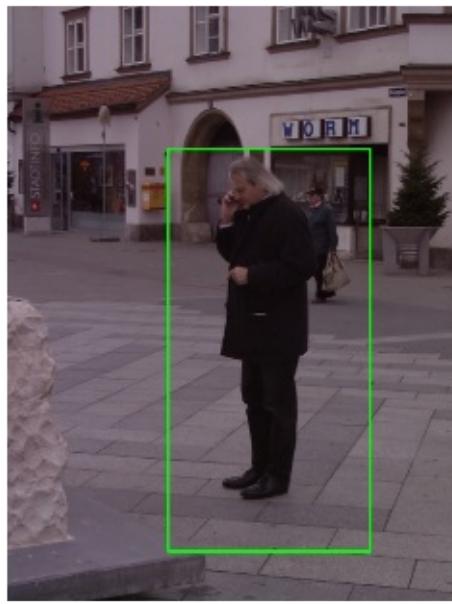


Figure 199: Example

### 17.18.0.9 Processing by Apache Spark

INRIA Person dataset provides 100+ images and Spark can be used for image processing in parallel. We load 288 images from “Test/pos” directory.

Spark provides a special object ‘sc’ to connect between a spark cluster and functions in python code. Therefore, we can run python functions in parallel to detect objects in this example.

- map function is used to process pedestrian and face detection per image from the parallelize() function of ‘sc’ spark context.
- collect function merges results in an array.

```
def apply_batch(imagePath): import cv2 import numpy as np
initialize the HOG descriptor/person detector hog =
cv2.HOGDescriptor()
hog.setSVMClassifier(cv2.HOGDescriptor_getDefaultPeopleDetector)
image = cv2.imread(imagePath) # detect people in the image
```

```
(rects, weights) = hog.detectMultiScale(image, winStride=(8, 8),
padding=(16, 16), scale=1.05) # draw the final bounding boxes
for (x, y, w, h) in rects: cv2.rectangle(image, (x, y), (x + w, y + h),
(0, 255, 0), 2)
return image
```

#### 17.18.0.9.1 Parallelize in Spark Context

The list of image files is given to parallelize.

```
pd = sc.parallelize(onlyfiles)
```

#### 17.18.0.9.2 Map Function (apply\_batch)

The 'apply\_batch' function that we created above is given to map function to process in a spark cluster.

```
pdc = pd.map(apply_batch)
```

#### 17.18.0.9.3 Collect Function

The result of each map process is merged to an array.

```
result = pdc.collect()
```

### 17.18.0.10 Results for 100+ images by Spark Cluster

```
for image in result:
 plt.figure()
 plt.axis("off")
 plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```



### ◆ Learning Objectives

- Learn quickly Go under the assumption you know a programming language
- Work with Go modules modules
- Contact some Go examples
- Learn about REST services in Go
- Learn how access virtual machines from Go
- Learn how to interface with kubernetes in Go



Go Logo

Go is a programming language that has been introduced by Google to replace the C++ language. Online documentation about go is available also from the official Go [Documentation \[86\]](#) Web page from which our material is derived.

The language Go has at its goal to be expressive, concise, clean, and efficient. It includes concurrency mechanisms with the goal to make it easy to write programs that can utilize multicore and networked features of modern computer systems and infrastructure easily with language features. However in contrast to languages such as python and ruby, it introduces in addition to static types explicitly types supporting concurrent programming such as channels that have already been used in the early days of programming for example as part of CSP [87] and OCCAM [88] [89].

In contrast to languages such as Python, Go is designed to compiled to machine code. However garbage collection and run-time reflection are built in, exposing this functionality similar to languages such as

python. Hence, it is designed to provide the programmer a fast, statically typed, compiled language that feels like a dynamically typed, interpreted language.

According to the [TIOBE \[90\]](#) index for programming languages Go has reached for November 2018 the 16th spot. However it is rated only with 1.081% with a declining rating but increase in the ranking. This trend is even more prominently depicted when looking at google trends in [Figure 200](#).

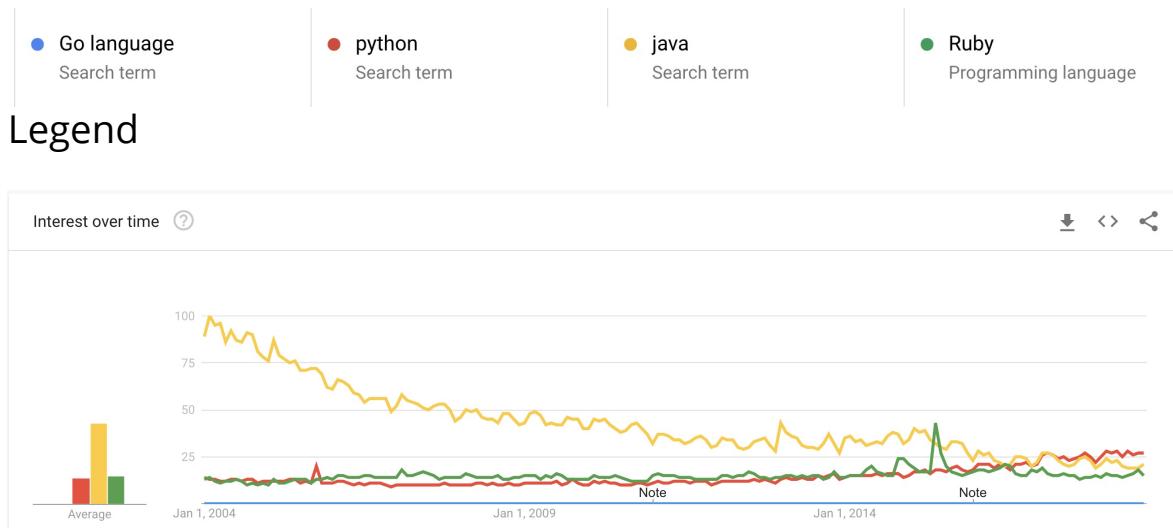


Figure 200: Google trends for selected programming languages

## 18.1 ORGANIZATION OF THE CHAPTER

The material presented in this chapter introduces the reader first to the basic concepts and features of the Go language and system. This includes instalation (see Section ?) and compiling (see Section ?), have a basic understanding of the programming language (see Section ?), use standard library and become familiar with package management (see Section ?). Next we will focus on aspects of the Go language that are especially useful for Cloud computing. This includes the review of how to develop REST services with various frameworks such as Gorilla (see Section ?) and OpenAPI (see Section ?). You will than be introduced on how to access virtual machines (see Section ?) and

containers (see Section ?).

In order to use Go we recommend that you have a computer fulfilling the following requirements:

- Have the most up to date version of Go installed
- Be familiar with the Linux command line as showcased in Section ?
- Familiarity with a text editor such as emacs, which we prefer as it supports nicely not only Go but any other language or document format we typically use in our activities. Alternatives are discussed in Section ?

## 18.2 REFERENCES

---

The following references may be useful for you to find out more about go. We have not gone to the list in detail, but want to make you aware of some of them that we found through simple searches in Google search. If you find others or you have a favourite, let us know and we will add them and mark them appropriately.

---

---

 TODO: the [?] are currently been worked on see [github](#)

---

- [golang.org](#) [86].
- [Go cheat sheet](#) [?].
- [The Little Go Book](#) [91].
- [Learn Go in an Hour - Video](#) 2015-02-15
- [Learning to Program in Go](#), a multi-part video training class [?].
- [Go By Example](#) provides a series of annotated code snippets [?].
- [Learn Go in Y minutes](#) is a top-to-bottom walk-through of the language [?].
- [Workshop-Go](#) - Startup Slam Go Workshop - examples and slides [?].

- [Go Fragments](#) - A collection of annotated Go code examples [?].
- [50 Shades of Go: Traps, Gotchas, Common Mistakes for New Golang Devs](#) [?]
- [Golang Tutorials](#) - A free online class [?].
- [The Go Bridge Foundry](#) [?] - A member of the [Bridge Foundry](#) [?] family, offering a complete set of free Go training materials with the goal of bringing Go to under-served communities.
- [Golangbot](#) - Tutorials to get started with programming in Go [?].
- [Algorithms to Go](#) - Texts about algorithms and Go, with plenty of code examples [92].
- [Go Language Tutorials](#) - List of Go sites, blogs and tutorials for learning Go language [?].
- [Golang Development Video Course](#) - A growing list of videos focused purely on Go development [?].

## 18.3 INSTALLATION



In case Go is not installed on your computer it is iease to install. Up to date informtion about the install process for your acitecture is available from <https://golang.org/doc/install>

We list now some brief instalation notes

We recommend that you use the tarball for Linux, and MacOS, that you can obtain for your platform here:

- <https://golang.org/dl/>

Once downloaded, unpack it with

```
$ sudo tar -C /usr/local -xzf go$VERSION.$OS-$ARCH.tar.gz
```

Packaged installers are also available for macOS, and Windows. They may provide you with the familiar platform specific instalation methods.

## **18.4 EDITORS SUPPORTING GO**

---



A large number of editor compatibilities and plugins are listed at

- <https://github.com/golang/go/wiki/IDEsAndTextEditorPlugins>

We recommend that you identify an editor form that list that will work for you. Due to the universality of emacs and its use for managing LaTeX as ewll as bibtex, we recommend that you use emacs, Important is that the editor supports lin breaks at the 80 character limit so the code is no=icely formatted for github. If your editor does not support this feature use emacs.

If your version of emacs does not yet have support for go, yo ucan find the go mode at

- [https://www.emacswiki.org/emacs/GoMode \[?\]](https://www.emacswiki.org/emacs/GoMode)

The documentation to it is provided at

- [http://dominik.honnef.co/posts/2013/03/writing\\_go\\_in\\_emacs/ \[?\]](http://dominik.honnef.co/posts/2013/03/writing_go_in_emacs/)

Other editors may include

- [GoLand](#) [?] which however in contrast to PyCharm CCommunity edition is not free. However as student and faculty one can get a free license via <https://www.jetbrains.com/student/>
- [Atom](#) [?]
- [vim](#) [?]

👉 Please help us complete this section while letting us know how each editor supports 80 chracter line wrap mode.

## **18.5 Go LANGUAGE**

---



Go is a computer language developed by Google with the goal to “build simple, reliable, and efficient software”. The language is open source and the main Web page is <https://golang.org/>

Go is specifically a systems-level programming language for large, distributed systems and highly-scalable network servers. It is meant to replace C++ and Java in terms of Google’s needs. Go was meant to alleviate some of the slowness and clumsiness of development of very large software systems.

- slow compilation and slow execution
- programmers that collaborate using different subsets of languages
- readability and documentation
- language consistency
- versioning issues
- multi-language builds
- dependencies being hard to maintain

The following program from the <https://golang.org/> web page shows the customary Hello World example:

```
package main

import "fmt"

func main() {
 /* This is a very easy program. */
 fmt.Println("Hello World!")
}
```

## 18.5.1 Concurrency in Go ○

### 18.5.1.1 GoRoutines (execution) ○

A GoRoutine in the Go programming language is a lightweight thread that is managed by Go runtime. If you just put ‘go’ before a function, it means that it will execute concurrently with the rest of the code.

○ insufficient explanation, no examples

### 18.5.1.2 Channels (communication)

Channels are pipes that connect concurrent GoRoutines. You are able to send values and signals over Channels from GoRoutine to GoRoutine. This allows for synchronizing execution.

 insufficient explanation, no examples, CSP reference missing

### 18.5.1.3 Select (coordination)

The Select statement in Go lets you wait and watch multiple operations on a channel. Combining GoRoutines and channels will show off the true power of concurrency in Go.

 insufficient explanation, no examples, CSP reference missing

## 18.6 LIBRARIES

---



Golang comes with a list of standard libraries in the following table, and more libraries can be found on this page: <https://golang.org/pkg/>

Name	Synopsis
archive	
tar	Package tar implements access to tar archives.
zip	Package zip provides support for reading and writing ZIP archives.
bufio	Package bufio implements buffered I/O. It wraps an io.Reader or io.Writer object, creating another object (Reader or Writer) that also implements the interface but provides buffering and some help for textual I/O.

builtin	Package builtin provides documentation for Go's predeclared identifiers.
bytes	Package bytes implements functions for the manipulation of byte slices.
compress	
bzip2	Package bzip2 implements bzip2 decompression.
flate	Package flate implements the DEFLATE compressed data format, described in RFC 1951.
gzip	Package gzip implements reading and writing of gzip format compressed files, as specified in RFC 1952.
lzw	Package lzw implements the Lempel-Ziv-Welch compressed data format, described in T. A. Welch, ``A Technique for High-Performance Data Compression'', Computer, 17(6) (June 1984), pp 8-19.
zlib	Package zlib implements reading and writing of zlib format compressed data, as specified in RFC 1950.
container	
heap	Package heap provides heap operations for any type that

---

	implements <code>heap.Interface</code> .
list	Package list implements a doubly linked list.
ring	Package ring implements operations on circular lists.
context	Package context defines the Context type, which carries deadlines, cancelation signals, and other request-scoped values across API boundaries and between processes.
crypto	Package crypto collects common cryptographic constants.
aes	Package aes implements AES encryption (formerly Rijndael), as defined in U.S. Federal Information Processing Standards Publication 197.
cipher	Package cipher implements standard block cipher modes that can be wrapped around low-level block cipher implementations.
des	Package des implements the Data Encryption Standard (DES) and the Triple Data Encryption Algorithm (TDEA) as defined in U.S. Federal Information Processing Standards Publication 46-3.
dsa	Package dsa implements the Digital Signature Algorithm, as defined in FIPS 186-3.

---

ecdsa	Package ecdsa implements the Elliptic Curve Digital Signature Algorithm, as defined in FIPS 186-3.
elliptic	Package elliptic implements several standard elliptic curves over prime fields.
hmac	Package hmac implements the Keyed-Hash Message Authentication Code (HMAC) as defined in U.S. Federal Information Processing Standards Publication 198.
md5	Package md5 implements the MD5 hash algorithm as defined in RFC 1321.
rand	Package rand implements a cryptographically secure random number generator.
rc4	Package rc4 implements RC4 encryption, as defined in Bruce Schneier's Applied Cryptography.
rsa	Package rsa implements RSA encryption as specified in PKCS#1.
sha1	Package sha1 implements the SHA-1 hash algorithm as defined in RFC 3174.
sha256	Package sha256 implements the SHA224 and SHA256 hash algorithms as defined in FIPS 180-4.

sha512	Package sha512 implements the SHA-384, SHA-512, SHA-512/224, and SHA-512/256 hash algorithms as defined in FIPS 180-4.
subtle	Package subtle implements functions that are often useful in cryptographic code but require careful thought to use correctly.
tls	Package tls partially implements TLS 1.2, as specified in RFC 5246.
x509	Package x509 parses X.509-encoded keys and certificates.
pkix	Package pkix contains shared, low level structures used for ASN.1 parsing and serialization of X.509 certificates, CRL and OCSP.
database	
sql	Package sql provides a generic interface around SQL (or SQL-like) databases.
driver	Package driver defines interfaces to be implemented by database drivers as used by package sql.
debug	
dwarf	Package dwarf provides access to DWARF debugging information loaded from executable files, as defined in the DWARF 2.0 Standard at <a href="http://dwarfstd.org/doc/dwarf-2.0.0.pdf">http://dwarfstd.org/doc/dwarf-2.0.0.pdf</a>

---

elf	Package elf implements access to ELF object files.
gosym	Package gosym implements access to the Go symbol and line number tables embedded in Go binaries generated by the gc compilers.
macho	Package macho implements access to Mach-O object files.
pe	Package pe implements access to PE (Microsoft Windows Portable Executable) files.
plan9obj	Package plan9obj implements access to Plan 9 a.out object files.
encoding	Package encoding defines interfaces shared by other packages that convert data to and from byte-level and textual representations.
ascii85	Package ascii85 implements the ascii85 data encoding as used in the btoa tool and Adobe's PostScript and PDF document formats.
asn1	Package asn1 implements parsing of DER-encoded ASN.1 data structures, as defined in ITU-T Rec X.690.
base32	Package base32 implements base32 encoding as specified by RFC 4648.

---

base64	Package base64 implements base64 encoding as specified by RFC 4648.
binary	Package binary implements simple translation between numbers and byte sequences and encoding and decoding of varints.
csv	Package csv reads and writes comma-separated values (CSV) files.
gob	Package gob manages streams of gobs - binary values exchanged between an Encoder (transmitter) and a Decoder (receiver).
hex	Package hex implements hexadecimal encoding and decoding.
json	Package json implements encoding and decoding of JSON as defined in RFC 7159.
pem	Package pem implements the PEM data encoding, which originated in Privacy Enhanced Mail.
xml	Package xml implements a simple XML 1.0 parser that understands XML name spaces.
errors	Package errors implements functions to manipulate errors.
expvar	Package expvar provides a standardized interface to public

---

expvar	variables, such as operation counters in servers.
flag	Package flag implements command-line flag parsing.
fmt	Package fmt implements formatted I/O with functions analogous to C's printf and scanf.
go	
ast	Package ast declares the types used to represent syntax trees for Go packages.
build	Package build gathers information about Go packages.
constant	Package constant implements Values representing untyped Go constants and their corresponding operations.
doc	Package doc extracts source code documentation from a Go AST.
format	Package format implements standard formatting of Go source.
importer	Package importer provides access to export data importers.
parser	Package parser implements a parser for Go source files.
printer	Package printer implements printing of AST nodes.
scanner	Package scanner implements a scanner for Go source text.

token	Package token defines constants representing the lexical tokens of the Go programming language and basic operations on tokens (printing, predicates).
types	Package types declares the data types and implements the algorithms for type-checking of Go packages.
hash	Package hash provides interfaces for hash functions.
adler32	Package adler32 implements the Adler-32 checksum.
crc32	Package crc32 implements the 32-bit cyclic redundancy check, or CRC-32, checksum.
crc64	Package crc64 implements the 64-bit cyclic redundancy check, or CRC-64, checksum.
fnv	Package fnv implements FNV-1 and FNV-1a, non-cryptographic hash functions created by Glenn Fowler, Landon Curt Noll, and Phong Vo.
html	Package html provides functions for escaping and unescaping HTML text.
template	Package template (html/template) implements data-driven templates for generating HTML output safe against code injection.

against code injection.

---

image	Package image implements a basic 2-D image library.
color	Package color implements a basic color library.
palette	Package palette provides standard color palettes.
draw	Package draw provides image composition functions.
gif	Package gif implements a GIF image decoder and encoder.
jpeg	Package jpeg implements a JPEG image decoder and encoder.
png	Package png implements a PNG image decoder and encoder.
index	
suffixarray	Package suffixarray implements substring search in logarithmic time using an in-memory suffix array.
io	Package io provides basic interfaces to I/O primitives.
ioutil	Package ioutil implements some I/O utility functions.
log	Package log implements a simple logging package.
syslog	Package syslog provides a simple interface to the system log service.

math	constants and mathematical functions.
big	Package big implements arbitrary-precision arithmetic (big numbers).
bits	Package bits implements bit counting and manipulation functions for the predeclared unsigned integer types.
cmplx	Package cmplx provides basic constants and mathematical functions for complex numbers.
rand	Package rand implements pseudo-random number generators.
mime	Package mime implements parts of the MIME spec.
multipart	Package multipart implements MIME multipart parsing, as defined in RFC 2046.
quotedprintable	Package quotedprintable implements quoted-printable encoding as specified by RFC 2045.
net	Package net provides a portable interface for network I/O, including TCP/IP, UDP, domain name resolution, and Unix domain sockets.
http	Package http provides HTTP client

http	and server implementations.
cgi	Package cgi implements CGI (Common Gateway Interface) as specified in RFC 3875.
cookiejar	Package cookiejar implements an in-memory RFC 6265-compliant http.CookieJar.
fcgi	Package fcgi implements the FastCGI protocol.
httptest	Package httptest provides utilities for HTTP testing.
httptrace	Package httptrace provides mechanisms to trace the events within HTTP client requests.
httputil	Package httputil provides HTTP utility functions, complementing the more common ones in the net/http package.
pprof	Package pprof serves via its HTTP server runtime profiling data in the format expected by the pprof visualization tool.
mail	Package mail implements parsing of mail messages.
rpc	Package rpc provides access to the exported methods of an object across a network or other I/O connection.
	Package jsonrpc implements a

ServerCodec for the rpc package.

---

smtp	Package smtp implements the Simple Mail Transfer Protocol as defined in RFC 5321.
textproto	Package textproto implements generic support for text-based request/response protocols in the style of HTTP, NNTP, and SMTP.
url	Package url parses URLs and implements query escaping.
os	Package os provides a platform-independent interface to operating system functionality.
exec	Package exec runs external commands.
signal	Package signal implements access to incoming signals.
user	Package user allows user account lookups by name or id.
path	Package path implements utility routines for manipulating slash-separated paths.
filepath	Package filepath implements utility routines for manipulating filename paths in a way compatible with the target operating system-defined file paths.
plugin	Package plugin implements loading and symbol resolution of

plugin	loading and symbol resolution of Go plugins.
reflect	Package reflect implements runtime reflection, allowing a program to manipulate objects with arbitrary types.
regexp	Package regexp implements regular expression search.
syntax	Package syntax parses regular expressions into parse trees and compiles parse trees into programs.
runtime	Package runtime contains operations that interact with Go's runtime system, such as functions to control goroutines.
cgo	Package cgo contains runtime support for code generated by the cgo tool.
debug	Package debug contains facilities for programs to debug themselves while they are running.
msan	
pprof	Package pprof writes runtime profiling data in the format expected by the pprof visualization tool.
race	Package race implements data race detection logic.

trace	for programs to generate traces for the Go execution tracer.
sort	Package sort provides primitives for sorting slices and user-defined collections.
strconv	Package strconv implements conversions to and from string representations of basic data types.
strings	Package strings implements simple functions to manipulate UTF-8 encoded strings.
sync	Package sync provides basic synchronization primitives such as mutual exclusion locks.
atomic	Package atomic provides low-level atomic memory primitives useful for implementing synchronization algorithms.
syscall	Package syscall contains an interface to the low-level operating system primitives.
js	Package js gives access to the WebAssembly host environment when using the js/wasm architecture.
testing	Package testing provides support for automated testing of Go packages.
	Package iotest implements

iotest	Package iotest implements Readers and Writers useful mainly for testing.
quick	Package quick implements utility functions to help with black box testing.
text	
scanner	Package scanner provides a scanner and tokenizer for UTF-8-encoded text.
tabwriter	Package tabwriter implements a write filter (tabwriter.Writer) that translates tabbed columns in input into properly aligned text.
template	Package template implements data-driven templates for generating textual output.
parse	Package parse builds parse trees for templates as defined by text/template and html/template.
time	Package time provides functionality for measuring and displaying time.
unicode	Package unicode provides data and functions to test some properties of Unicode code points.
utf16	Package utf16 implements encoding and decoding of UTF-16 sequences.

utf8	functions and constants to support text encoded in UTF-8.
unsafe	Package unsafe contains operations that step around the type safety of Go programs.

## 18.7 Go CMD



## 18.7.1 CMD

In Python we have the `CMD5` package that allows us to create command shells with plugins. In Go we find a community developed package called `gosh` (or Go shell). It uses the Go plugin system to create interactive console-based shell programs. A shell created with `gosh` contains a collection of Go plugins, each of which implement one or more commands. Upon start `gosh` starts, searches the directory `./plugins` and loads them so they become available within `gosh`.

- <https://github.com/vladimirvivien/gosh>

## 18.7.2 DocOpt

When we want to design commandline arguments for go programs we have many options. However, as our approach is to create documentation first, docopts provides also a good approach for Go. The code for it is located at

- <https://github.com/docopt/docopt.go>

It can be installed with

```
$ go get github.com/docopt/docopt-go
```

A sample programs are located at

- <https://github.com/docopt/docopt.go/blob/master/examples/>

A sample program of using doc opts for our purposes looks as follows.

```
package main

import (
 "fmt"
 "github.com/docopt/docopt-go"
)

func main() {
 usage := `cm-go.

Usage:
 cm-go vm start NAME [--cloud=CLOUD]
 cm-go vm stop NAME [--cloud=CLOUD]
 cm-go set --cloud=CLOUD
 cm-go -h | --help
 cm-go --version

Options:
 -h --help Show this screen.
 --version Show version.
 --cloud=CLOUD The name of the cloud.
 --moored Moored (anchored) mine.
 --drifting Drifting mine.

ARGUMENTS:
 NAME The name of the VM`

 arguments, _ := docopt.ParseDoc(usage)
 fmt.Println(arguments)
}
```

## 18.8 Go REST



Go is a new powerful language and there are many frameworks from lightweight to full featured that support building RESTful APIs.

1. [Revel](#) A high-productivity web framework for the Go language.
2. [Gin](#) The fastest full-featured web framework for Golang. Crystal clear.
3. [Martini](#) Classy web framework for Go
4. [Web.go](#) The easiest way to create web applications with Go

List here the rest services tutorials for frameworks

- <https://nordicapis.com/7-frameworks-to-build-a-rest-api-in-go/>
- with mongo <https://hackernoon.com/build-restful-api-in-go-and-mongodb-5e7f2ec4be94>
- <https://tutorialedge.net/golang/consuming-restful-api-with-go/>
- <https://thenewstack.io/make-a-restful-json-api-go/>
- Making a RESTful JSON API in Go

### **18.8.1 Gorilla**

- <https://www.codementor.io/codehakase/building-a-restful-api-with-golang-a6yivzqdo>

Gorilla is a web toolkit for the Go programming language. Currently these packages are available:

- gorilla/context stores global request variables.
- gorilla/mux is a powerful URL router and dispatcher.
- gorilla/reverse produces reversible regular expressions for regexp-based muxes.
- gorilla/rpc implements RPC over HTTP with codec for JSON-RPC.
- gorilla/schema converts form values to a struct.
- gorilla/securecookie encodes and decodes authenticated and optionally encrypted cookie values.
- gorilla/sessions saves cookie and filesystem sessions and allows custom session backends.
- gorilla/websocket implements the WebSocket protocol defined in RFC 6455.

### **18.8.2 REST, RESTful**

REST is an acronym for Representational State Transfer. It is a web standards architecture and HTTP Protocol. The REST protocol, describes six (6) constraints:

1. Uniform Interface
2. Cacheable
3. Client-Server
4. Stateless
5. Code on Demand
6. Layered System

### 18.8.3 Router

Package gorilla/mux implements a request router and dispatcher for matching incoming requests to their respective handler.

The name mux stands for “HTTP request multiplexer”. Like the standard `http.ServeMux`, `mux.Router` matches incoming requests against a list of registered routes and calls a handler for the route that matches the URL or other conditions. The main features are:

We'll need to use a mux to route requests, so we need a Go package for that (mux stands for HTTP request multiplexer which matches an incoming request to against a list of routes (registered)). In the rest-api directory, let's require the dependency (package rather). More examples are here: <https://github.com/gorilla/mux#examples>

```
rest-api$ go get github.com/gorilla/mux
```

```
package main

import (
 "encoding/json"
 "log"
 "net/http"
 "github.com/gorilla/mux"
)

// our main function
func main() {
 router := mux.NewRouter()
 router.HandleFunc("/people", GetPeople).Methods("GET")
 router.HandleFunc("/people/{id}", GetPerson).Methods("GET")
 router.HandleFunc("/people/{id}", CreatePerson).Methods("POST")
 router.HandleFunc("/people/{id}", DeletePerson).Methods("DELETE")
 log.Fatal(http.ListenAndServe(":8000", router))
}
```

Packages are explained here: \* `fmt` is what we'll be using to print to

STDOUT (the console) \* log is used to log when the server exits \* encoding/json is for creating our JSON responses \* net/http will give us the representations of HTTP requests, responses, and be responsible for running our server \* github.com/gorilla/mux will be our router that will take requests and decide what should be done with them

## 18.8.4 Full code

```
package main

import (
 "encoding/json"
 "github.com/gorilla/mux"
 "log"
 "net/http"
)

// The person Type (more like an object)
type Person struct {
 ID string `json:"id,omitempty"`
 Firstname string `json:"firstname,omitempty"`
 Lastname string `json:"lastname,omitempty"`
 Address *Address `json:"address,omitempty"`
}
type Address struct {
 City string `json:"city,omitempty"`
 State string `json:"state,omitempty"`
}

var people []Person

// Display all from the people var
func GetPeople(w http.ResponseWriter, r *http.Request) {
 json.NewEncoder(w).Encode(people)
}

// Display a single data
func GetPerson(w http.ResponseWriter, r *http.Request) {
 params := mux.Vars(r)
 for _, item := range people {
 if item.ID == params["id"] {
 json.NewEncoder(w).Encode(item)
 return
 }
 }
 json.NewEncoder(w).Encode(&Person{})
}

// create a new item
func CreatePerson(w http.ResponseWriter, r *http.Request) {
 params := mux.Vars(r)
 var person Person
```

```

 _ = json.NewDecoder(r.Body).Decode(&person)
 person.ID = params["id"]
 people = append(people, person)
 json.NewEncoder(w).Encode(people)
 }

// Delete an item
func DeletePerson(w http.ResponseWriter, r *http.Request) {
 params := mux.Vars(r)
 for index, item := range people {
 if item.ID == params["id"] {
 people = append(people[:index], people[index+1:]...)
 break
 }
 }
 json.NewEncoder(w).Encode(people)
}

// main function to boot up everything
func main() {
 router := mux.NewRouter()
 people = append(people, Person{ID: "1", Firstname: "John", Lastname: "Doe", Address: &Address{City:
 people = append(people, Person{ID: "2", Firstname: "Koko", Lastname: "Doe", Address: &Address{City:
 router.HandleFunc("/people", GetPeople).Methods("GET")
 router.HandleFunc("/people/{id}", GetPerson).Methods("GET")
 router.HandleFunc("/people/{id}", CreatePerson).Methods("POST")
 router.HandleFunc("/people/{id}", DeletePerson).Methods("DELETE")
 log.Fatal(http.ListenAndServe(":8000", router))
}

```

## 18.9 OPEN API O ?



⚠ We have a large section previously on openapi, what needs to be done here is to showcase how to generate go from swagger codegen or other tool and use it

### 18.9.1 serve specification UI

Most basic use-case: serve a UI for your spec:

```
swagger serve https://raw.githubusercontent.com/swagger-api/swagger-spec/master/examples/v2/swagger.json
```

### 18.9.2 validate a specification

```
swagger validate https://raw.githubusercontent.com/swagger-api/swagger-spec/master/examples/v2/swagger.json
```

### 18.9.3 Generate a Go OpenAPI server

⚠️ this is incorrect ○

```
swagger generate server [-f ./swagger.json] -A [application-name] [--principal [principal-na
◀ ▶]
```

### 18.9.4 generate a Go OpenAPI client

⚠️ this is incorrect ○

```
swagger generate client [-f ./swagger.json] -A [application-name] [--principal [principal-na
◀ ▶]
```

### 18.9.5 generate a spec from the source

⚠️ this is put here without explanation

```
swagger generate spec -o ./swagger.json
```

### 18.9.6 generate a data model

⚠️ this is put here without explanation

```
swagger generate model --spec={spec}
```

### 18.9.7 other editors

- [KaiZen-OpenAPI-Editor](#) - Full-featured Eclipse editor for OpenAPI 2.0 and 3.0, also available on Eclipse Marketplace.
- [Atom/linter-swagger](#) - This plugin for Atom Linter will lint Swagger 2.0 specifications or OpenAPI 3.0, both JSON and YAML using swagger-parser node package.
- [Swagger Editor](#) - Design, describe, and document your API on the first open source editor fully dedicated to OpenAPI-based APIs.
- [RepreZen API Studio](#) - RepreZen API Studio is an integrated workbench that brings API-first design into focus for your whole team, harmonizes your API designs, and generates APIs that click into client apps.

- [Apicurio Studio](#) A standalone API design studio that can be used to create new or edit existing API designs.
- [SwaggerHub](#) - API design and documentation platform to improve collaboration, standardize development workflow and centralize their API discovery and consumption.
- [Senya Editor](#) - Design API specifications fast and effectively in your favorite JetBrains IDE.

## 18.9.8 References

- [go-swagger documentation](#)
- [OpenAPI.Tools](#)

## 18.10 Go CLOUD O ?



### 18.10.1 Golang Openstack Client

- <https://github.com/openstack/golang-client>

Examples:

- [Authentication](#)
- [Images](#)
- [ObjectStore](#)
- This file reads in some configurations from a json file, however we want to develop one for our `~/.cloudmesh/cloudmesh.yaml` file. For the json example see: [json setup](#)
- [Volume](#)

Portable Cloud Programming with Go Cloud:

- <https://blog.golang.org/go-cloud>

### 18.10.2 OpenStack from Go

There are multiple API interfaces that allow direct access to elementary functionality to control for example virtual machines in

Go. This includes GopherCloud, and GolangClient. We describe them next.

### 18.10.2.1 GopherCloud

GopherCloud is located at

- <https://github.com/gophercloud/gophercloud>

#### 18.10.2.1.1 Authentication

To interact with OpenStack, you will need to first authenticate with your OpenStack cloud. You will need to know your username and password. However the example that we provide here is intentionally wrong to showcase you a better way. The example includes a hard coded username and password, that actually is supposed to be either read in via an interactive process or from a `~/.cloudmesh/cloudmesh.yaml` file as we have used in our python cloudmesh code. We will use the same format and obtain the information from that file. Your task will be to write a yaml file reader in go, get the information and modify the program accordingly while improving this section with a pull request.

The example copied from gopher cloud looks as follows:

```
import (
 "github.com/gophercloud/gophercloud"
 "github.com/gophercloud/gophercloud/openstack"
 "github.com/gophercloud/gophercloud/openstack/utils"
)

opts := gophercloud.AuthOptions{
 IdentityEndpoint: "https://openstack.example.com:5000/v2.0",
 Username: "{username}",
 Password: "{password}",
}
```

Naturally you can also obtain the values from environment variables as pointed out by gopher cloud:

```
import (
 "github.com/gophercloud/gophercloud"
 "github.com/gophercloud/gophercloud/openstack"
 "github.com/gophercloud/gophercloud/openstack/utils"
)
```

```
opts, err := openstack.AuthOptionsFromEnv()
provider, err := openstack.AuthenticatedClient(opts)
```

We will at this time not use the second approach.

To start a virtual machine you need to first identify the location of the client for the region you will use. This can be achieved with the command:

```
client, err := openstack.NewComputeV2(provider, gophercloud.EndpointOpts{
 Region: os.Getenv("OS_REGION_NAME"),
})
```

#### 18.10.2.1.2 Virtual machines

Now that we know we can authenticate to the cloud, we can create our first virtual machine, where `flavor_id` and `image_id` are the appropriate flavors and image ids:

```
import "github.com/gophercloud/gophercloud/openstack/compute/v2/servers"

server, err := servers.Create(client, servers.CreateOpts{
 Name: "gregor-001",
 FlavorRef: "flavor_id",
 ImageRef: "image_id",
}).Extract()
```

Additional information can be found in the source code of GoherClient which you can easily inspect. Some useful documentation is also provided in <https://github.com/gophercloud/gophercloud/blob/master/doc.go>

#### 18.10.2.1.3 Resources

Code examples are provided from <https://github.com/gophercloud/gophercloud/blob/master/doc.go>

As OpenStack is providing REST interfaces, gopher cloud leverages this model. Hence, it provides interfaces to manage REST resources. These resources are bound to structs so they can easily be manipulated and interfaced with. To for example get the client with a specific `{serverId}` and extract its information we can use the following API call:

```
server, err := servers.Get(client, "{serverId}").Extract()
```

If we need just a subset of the information, we can get an intermediate result with just the get method. Then we can obtain specific information from the result as needed.

```
result := servers.Get(client, "{serverId}")
// Attempt to extract the disk configuration from the OS-DCF disk config
// extension:
config, err := diskconfig.ExtractGet(result)
```

The previous example is based on a single resource. However, if we interact with a list of resources we need to use the `Pager` struct so we can iterate over each page. A convenient example is provided next. Here we list all servers while iterating over all pages returned to us. While calling each page we can invoke special operations that are applied to each page.

```
err := servers.List(client, nil).EachPage(func (page pagination.Page) (bool, error) {
 s, err := servers.ExtractServers(page)
 if err != nil {
 return false, err
 }
 // Handle the []servers.Server slice.
 // Return "false" or an error to prematurely stop fetching new pages.
 return true, nil
})
```

However, if we just want to provide a list of all servers, we can simply use the `AllPages()` method as follows:

```
allPages, err := servers.List(client, nil).AllPages()
allServers, err := servers.ExtractServers(allPages)
```

Additional methods and resources can be found at

- <https://github.com/gophercloud/gophercloud/blob/master/do>

## 18.11 Go Links



Section provided by Gregor

- [https://cse.sc.edu/~mgv/csce330f16/pres/330f15\\_BarnhartRee](https://cse.sc.edu/~mgv/csce330f16/pres/330f15_BarnhartRee)
- <http://www.cis.upenn.edu/~matuszek/cis554->

- [2016/Talks/Golang\\_Presentation.ppt](2016/Talks/Golang_Presentation.ppt)
- <https://talks.golang.org/2015/go-for-java-programmers.slide#1>
- <https://github.com/golang/go/wiki/GoTalks>
- <http://courses.cs.vt.edu/cs5204/fall11-kafura/Overheads/Go.pptx>
- [https://en.wikipedia.org/wiki/Kahn\\_process\\_networks](https://en.wikipedia.org/wiki/Kahn_process_networks)

## 18.11.1 Languages

- <http://devcodegeek.com/best-cloud-programming-languages.html>
- <https://webdesignledger.com/top-4-cloud-computing-languages-learn-now/>
- <https://techlog360.com/top-10-cloud-programming-languages/>
- <https://www.techrepublic.com/article/10-of-the-coolest-cloud-programming-languages/>

## 18.11.2 Popular

- <http://www.zdnet.com/article/which-programming-languages-are-most-popular-and-what-does-that-even-mean/>
- <https://www.tiobe.com/tiobe-index/>

## 18.11.3 PYPL Popularity of Programming Language

``The PYPL PopularitY of Programming Language Index is created by analyzing how often language tutorials are searched on Google.

The more a language tutorial is searched, the more popular the language is assumed to be. It is a leading indicator. The raw data comes from Google Trends."

- <http://pypl.github.io/PYPL.html>
- <https://www.infoworld.com/article/2610933/cloud->

[computing/article.html](#)

#### **18.11.4 Specification to Program**

- <https://swagger.io/>
- <https://nordicapis.com/top-specification-formats-for-rest-apis/>
- <https://www.npmjs.com/package/raml-python>
- <https://github.com/Jumpscale/go-raml>
- [https://en.wikipedia.org/wiki/Overview\\_of\\_RESTful\\_API\\_Descri](https://en.wikipedia.org/wiki/Overview_of_RESTful_API_Descri)

#### **18.11.5 Heroku and Go**

TBD



⚠ Although we do not intend to use julia in this class, we provide the option for a small number of students to develop this chapter as part of the course requirements. In case you chose it, you could also investigate how to use the cloud from it as to potentially do your project in julia. Your project would be to develop a cloud client in julia similar to the one we developed in python under the name cloudmesh.

This chapter can be contributed by one or multiple students

- <https://julialang.org/>

## **19.1 LANGUAGE**

---

Include here basic language features

## **19.2 PARALLEL LANGUAGE CONSTRUCTS**

---

Include here parallel language constructs

- Missing Values
- Network Streams
- Parallel Computing

## **19.3 INTERFACEING WITH THE SYSTEM**

---

- External programs
- Environment variables

## **19.4 REST IN JULIA**

---

TBD

## **19.5 OPENSTACK IN JULIA**

---

TBD

## **19.6 AWS IN JULIA**

---

TBD

## **19.7 AZURE IN JULIA**

---

TBD

## **19.8 FAAS IN JULIA**

---



⚠ This section will not be taught as part of this class, However interested students can contribute a chapter. This section could be contributed to, Focus on the aspect cloud computing and not just on the language features of scala.

Links not read just googles scala and cloud:

- <http://www.opencirrus.org/scala-java-cloud/>
- <https://www.theserverside.com/feature/Scala-and-the-Cloud>
- <https://cloud.google.com/dataproc/docs/tutorials/spark-scala>
- <https://dl.acm.org/citation.cfm?id=1863103.1863113>

REST:

- <https://nordicapis.com/8-frameworks-to-build-a-web-api-in-scala/>



### 21.1 MQTT



#### 🎓 Learning Objectives

- Understand Message systems for the cloud
- Learn about MQTT

With the increase importance of cloud computing and the increased number of edge devices and their applications, such as sensor networks, it is crucial to enable fast communication between the sensing devices and actuators, which may not be directly connected, as well as cloud services that analyze the data. To allow services that are built on different software and hardware platforms to communicate, a data agnostic, fast and reliable service is needed. In addition to communication, the data generated by these devices, services, and sensors must be analyzed. Security aspects to relay this data is highly important. We will introduce a service called MQTT, which is a common, easy to use, queuing protocol that helps meet these requirements.

#### 21.1.1 Introduction

As Cloud Computing and Internet of Things (IoT) applications and sensor networks become commonplace and more and more devices are being connected, there is an increased need to allow these devices to communicate quickly and securely. In many cases these edge devices have very limited memory and need to conserve power. The computing power on some of these devices is limited so that the sensory data need to be analyzed remotely. Furthermore, they may not even have enough computing capacity to process traditional HTTP web requests efficiently [93][94] or these traditional Web-based

services are too resource hungry. Monitoring the state of a remotely located sensor using HTTP would require sending requests and receiving responses to and from the device frequently, which may not be efficient on small circuits or embedded chips on edge computing sensors [93].

Message Queue Telemetry Transport (MQTT) is a lightweight machine to machine (M2M) messaging protocol, based on a client/server publish-subscribe model. It provides a simple service allowing us to communicate between sensors, and services based on a subscription model.

MQTT was first developed in 1999 to connect oil pipelines [94]. The protocol has been designed to be used on top of TCP/IP protocol in situations where network bandwidth, and available memory are limited allowing low power usage. However, as it is implemented on top of TCP/IP it is reliable in contrast to other protocols such as UDP. It allows efficient transmission of data to various devices listening for the same event, and is scalable as the number of devices increase [95][96].

MQTT is becoming more popular than ever before with the increasing use of mobile device and smartphone applications such as Facebook's Messenger and Amazon Web Services. This protocol is used in WIFI or low bandwidth network. MQTT does not require any connection with the content of the message.

The current support for MQTT is conducted as part of the Eclipse Phao project [97]. As MQTT is a protocol many different clients in various languages exist. This includes languages such as Python, C, Java, Lua, and many more.

The current standard of MQTT is available at

<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf>

## 21.1.2 Publish Subscribe Model

MQTT works via a publish-subscribe model that contains 3 entities: (1) a Raspberry Pi publisher, that sends a message, (2) a broker, that maintains queue of all messages based on topics and (3) multiple subscribers that subscribe to various topics they are interested in [98]. See [Figure 201](#)

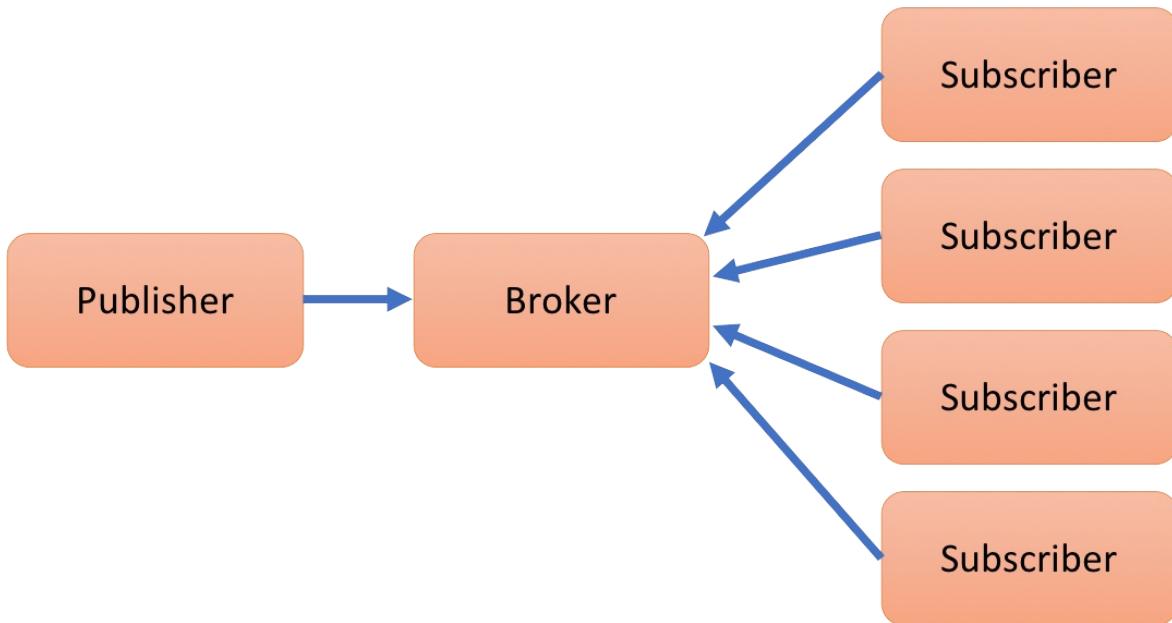


Figure 201: MQTT publish subscriber model

This allows for decoupling of functionality at various levels. The publisher and subscriber do not need to be close to each other and do not need to know each others identity. They need only to know the broker, as the publisher and the subscribers do not have to be running either at the same time nor on the same hardware [99].

Ready to use implementation exist to be deployed as brokers in the users application frameworks. A broker is a service that relays information between the client and servers. Common brokers include the open source Mosquito broker [96] and the Eclipse Phao MQTT Broker [97].

### 21.1.2.1 Topics

MQTT implements a hierarchy of topics that are relates to all messages. These topics are recognised by strings separated by a

forward-slash (/), where each part represents a different topic level. This is a common model introduced in file systems but also in internet URLs.

A topic looks therefore as follows:

```
topic-level0/topic-level1/topic-level2.
```

Subscribers can subscribe to different topics via the broker. Subscribing to `topic-level0` allows the subscriber to receive all messages that are associated with topics that start with `topic-level0`. This allows subscribers to filter what messages to receive based on the topic hierarchy. Thus, when a publisher publishes a message related to a topic to the broker, the message is forwarded to all the clients that have subscribed to the topic of the message or a topic that has a lower depth of hierarchy [99] [98].

This is different from traditional point-to-point message queues as the message is forwarded to multiple subscribers, and allows for flexibility of dealing with subscribed topics not only on the server but also on the subscriber side [99]. The basic steps in a MQTT client subscriber application include to (1) connect to the broker, (2) subscribe to some topics, (3) wait for messages and (4) perform the appropriate action when a certain message is received [95].

### 21.1.2.2 Callbacks

One of the advantages of using MQTT is that it supports asynchronous behaviour with the help of callbacks. Both the publisher and subscriber can use non-blocking callbacks to act upon message exchanges. [99][100].

For example, the paho-mqtt package for python provides callbacks methods including `on-connect()`, `on-message()` and `on-disconnect()`, which are invoked when the connection to the broker is complete, a message is received from the broker, and when the client is disconnected from the broker respectively. These methods are used in conjunction with the `loop-start()` and `loop-end()` methods which start and end an

asynchronous loop that listens for these events invoking the relevant callbacks. Hence it frees the services to perform other tasks [100] when no messages are available, thus reducing overhead.

### 21.1.2.3 Quality of Service

MQTT has been designed to be flexible allowing for the change of quality of service (QoS) as desired by the application. Three basic levels of QoS are supported by the protocol: Atmost-once (QoS level 0), Atleast-once (QoS level 1) and Atmost-once (QoS level 2) [100], [101].

QoS level 0:

The QoS level of 0 is used in applications where some dropped messages may not affect the application. Under this QoS level, the broker forwards a message to the subscribers only once and does not wait for any acknowledgement [101][100].

QoS Level 1:

The QoS level of 1 is used in situations where the delivery of all messages is important and the subscriber can handle duplicate messages. Here the broker keeps on resending the message to a subscriber after a certain timeout until the first acknowledgement is received.

QoS Level 3:

A QoS level of 3 is used in cases where all messages must be delivered and no duplicate messages should be allowed. In this case the broker sets up a handshake with the subscriber to check for its availability before sending the message [100], [101].

The various levels of quality of service allow the use of this protocol with different service level expectations.

### 21.1.3 Secure MQTT Services

MQTT specification uses TCP/IP to deliver the messages to the subscribers, but it does not provide security by default to enable resource constrained IoT devices. “It allows the use of username and password for authentication, but by default this information is sent as plain text over the network, making it susceptible to man-in-the middle attacks” [102], [103]. Therefore, to support sensitive applications additional security measures need to be integrated through other means. This may include for example the use of Virtual Private Networks (VPNs), Transport Layer Security, or application layer security [103].

#### **21.1.3.1 Using TLS/SSL**

Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are cryptographic protocols that establish the identity of the server and client with the help of a handshake mechanism which uses trust certificates to establish identities before encrypted communication can take place [104]. If the handshake is not completed for some reason, the connection is not established and no messages are exchanged [103]. “Most MQTT brokers provide an option to use TLS instead of plain TCP and port 8883 has been standardized for secured MQTT connections” [102].

Using TLS/SSL security however comes at an additional cost. If the connections are short-lived then most of the time is spent in verifying the security of the handshake itself, which in addition to using time for encryption and decryption, may take up few kilobytes of bandwidth. In case the connections are short-lived, temporary session IDs and session tickets can be used as alternative to resume a session instead of repeating the handshake process. If the connections are long term, the overhead of the handshake is negligible and TLS/SSL security should be used [102], [103].

#### **21.1.3.2 Using OAuth**

OAuth is an open protocol that allows access to a resource without providing unencrypted credentials to the third party. Although MQTT

protocol itself does not include authorization, many MQTT brokers include authorization as an additional feature [104]. OAuth2.0 uses JSON Web Tokens which contain information about the token and the user and are signed by a trusted authorization server [105].

When connecting to the broker this token can be used to check whether the client is authorised to connect at this time or not. Additionally the same validations can be used when publishing or subscribing to the broker. The broker may use a third party resource such as LDAP (lightweight directory access protocol) to look up authorizations for the client [105]. Since there can be a large number of clients and it can become impractical to authorize everyone, clients may be grouped and the authorizations may be checked for each group [104].

#### **21.1.4 Integration with Other Services**

As the individual IoT devices perform their respective functions in the sensor network, a lot of data is generated which needs to be processed. MQTT allows easy integration with other services, that have been designed to process this data. Let us provide some examples of MQTT integration into other Services.

Apache Storm.

Apache storm is a distributed processing system that allows real time processing of continuous data streams, much like Hadoop works for batch processing [106]. Apache storm can be easily integrated with MQTT as shown in [107] to get real time data streams and allow analytics and online machine learning in a fault tolerant manner [108].

ELK stack.

ELK stack (elastic-search, logstash and kibana) is an opensource project designed for scalability which contains three main software packages, the elastic-search search and analytics engine, logstash which is a data collection pipeline and kibana

which is a visualization dashboard [109]. Data from an IoT network can be collected, analysed and visualized easily with the help of the ELK stack as shown in [110] and [111].

## 21.1.5 MQTT in Production

When using optimized MQTT broker services, MQTT can be utilized for enterprise and production environments. A good example is the use of EMQ (Erlang MQTT Broker) that provides a highly scalable, distributed and reliable MQTT broker for enterprise-grade applications [112].

## 21.1.6 Installation

The installation of an MQTT server based on paho is very simple.

### 21.1.6.1 MacOS install

On OSX yo need to first install mosquito, which is easiest to install with `brew`

Step 1: Installing Mosquito clients

Open a terminal and use homebrew to install mosquito and than you can install paho with pip

```
brew install mosquitto
pip install paho-mqtt
```

You need to start the mosquito service buy hand to use it.

### 21.1.6.2 MacOS Advanced Service install

⚠ We recommend that this is only be done if you truly need a production system. For our class you will not need this.

You can integrate mosquito service on boot, while adding it via LaunchAgents. This can be achieved by linking it as follows:

```
ln -sfv /usr/local/opt/mosquitto/*.plist ~/Library/LaunchAgents
```

Next you need to restart the server as follows:

```
launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mosquitto.plist
```

Now you can test the installation and ensure the server is running successfully. Open a new command window and start a listener.

```
mosquitto_sub -t topic/state
```

To test the setup you can in another window, send a message to the listener.

```
mosquitto_pub -t topic/state -m "Hello World"
```

This ensures the server is running.

### 21.1.6.3 Ubuntu install

On ubuntu you need to first install mosquito, than with pip you install paho-mqt

```
$ sudo apt-get install mosquitto mosquitto-clients
$ pip install paho-mqtt
```

### 21.1.6.4 Raspberry Pi Setup

#### 21.1.6.4.1 Broker

You will need to add the mosquito repository to the known repositories as follows:

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
apt-get update
```

Mosquito is installed by implementing the following command:

```
apt-get install mosquito
```

#### 21.1.6.4.2 Client

The MQTT client needs to be installed on raspberry pi by running the following command:

```
apt-get install mosquitto-clients
```

## 21.1.7 Server UseCase

In this example we are demonstrating how to set up a MQTT broker, a client and a subscriber while just using regular servers and clients. The code of this example is located at:

- <https://github.com/bigdata-i523/sample-hid000/tree/master/experiment/mqtt>

A test program that starts a MQTT broker and client showcases how simple the interactions between the publisher and subscribers are while using a higher level API such as provided through the python client library of Paho.

```
import paho.mqtt.client as mqtt
import time

def on_message(client, userdata, message):
 print("message received ",
 str(message.payload.decode("utf-8")))
 print("message topic=", message.topic)
 print("message qos=", message.qos)
 print("message retain flag=", message.retain)

def on_log(client, userdata, level, buf):
 print("log: ",buf)

broker_address="localhost"
broker_address="test.mosquitto.org"
broker_address="broker.hivemq.com"
broker_address="iot.eclipse.org"

print("creating new instance")
client = mqtt.Client("i523") #create new instance
client.on_log=on_log
client.on_message=on_message #attach function to callback

print("connecting to broker")
client.connect(broker_address) #connect to broker
client.loop_start() #start the loop

print("Subscribing to topic","robot/leds/led1")
```

```

client.subscribe("robot/leds/led1")

print("Publishing message to topic", "robot/leds/led1")
client.publish("robot/leds/led1", "OFF")

time.sleep(4) # wait
client.loop_stop() #stop the loop

```

## 21.1.8 IoT Use Case with a Raspberry PI

MQTT can be used in a variety of applications. This section explores a particular use case of the protocol. A small network was set up with three devices to simulate an IoT environment, and actuators were controlled with the help of messages communicated over MQTT.

The code for the project is available at

- <https://github.com/bigdata-i523/hid201/tree/master/experiment/mqtt>

### 21.1.8.1 Requirements and Setup

The setup used three different machines. A laptop or a desktop running the MQTT broker, and two raspberry pis configured with raspbian operating system. Eclipse Paho MQTT client was setup on each of the raspberry pis [100]. Additionally all three devices were connected to an isolated local network.

GrovePi shields for the raspberry pis, designed by Dexter Industries were used on each of the raspberry pis to connect the actuators as they allow easy connections ot the raspberry pi board [113]. The actuators used were Grove relays [114] and Grove LEDs [115] which respond to the messages received via MQTT.

To control the leds and relays, the python library cloudmesh.pi [116], developed at Indiana University was used. The library consists of interfaces for various IoT sensors and actuators and can be easily used with the grove modules.

### 21.1.8.2 Results

The two Raspberry Pis subscribe connect to the broker and subscribe with different topics. The raspberry pis wait for any messages from the broker. A publisher program that connects to the broker publishes messages to the broker for the topics that the two raspberry pis had registered. Each raspberry pi receives the corresponding message and turns the LEDs or relays on or off as per the message.

On a local network this process happens in near real time and no delays were observed. Eclipse IoT MQTT broker ([iot.eclipse.org](http://iot.eclipse.org)) was also tried which also did not result in any significant delays.

Thus it is observed that two raspberry pis can be easily controlled using MQTT. This system can be extended to include arbitrary number of raspberry pis and other devices that subscribe to the broker. If a device fails, or the connection from one device is broken, other devices are not affected and continue to perform the same.

This project can be extended to include various other kinds of sensors and actuators. The actuators may subscribe to topics to which various sensors publish their data and respond accordingly. The data of these sensors can be captured with the help of a data collector which may itself be a different subscriber, that performs analytics or visualizations on this data.

### **21.1.9 Conclusion**

We see that as the number of connected devices increases and their applications become commonplace, MQTT allows different devices to communicate with each other in a data agnostic manner. MQTT uses a publish-subscribe model and allows various levels of quality of service requirements to be fulfilled. Although MQTT does not provide data security by default, most brokers allow the use of TLS/SSL to encrypt the data. Additional features may be provided by the broker to include authorization services. MQTT can be easily integrated with other services to allow collection and analysis of data. A small environment was simulated that used MQTT broker and clients

running on raspberry pis to control actuators

### **21.1.10 Exercises**

E.MQTT.1:

Develop a temperature broker, that collects the temperature from a number of machines and clients can subscribe to the data and visualize it.

E.MQTT.2:

Develop a CPU load broker, that collects the cpu load from a number of machines and clients can subscribe to the data and visualize it.

E.MQTT.3:

Develop a broker with a variety of topics that collects data from a Raspberry Pi or Raspberry PI cluster and visualize it.

E.MQTT.4:

Explore hosted services for MQTT while at the same time remembering that they could pose a security risk. Can any of the online services be used to monitor a cluster safely?

## **21.2 GRAPHQL**

---



### **🎓 Learning Objectives**

- Learn about GraphQL
  - Develop a GraphQL Server in Python
-

## 21.2.1 Introduction

GraphQL is a data query language developed by Facebook.

GraphQL allows clients to request they need while specifying attributes in the query without thinking much about the API implementation. It simplifies access and reduces traffic as the application has control over the data it needs and its format. Hence GraphQL reduces the network traffic as only the necessary data is transferred from server to client.

Unlike REST APIs, which require often loading data via multiple queries, GraphQL can get typically all the data in a single request. GraphQL APIs are defined in terms of types and fields. Types help GraphQL to ensure that client only asks for what is possible and in case of faults, provides clear and helpful errors.

Initially GraphQL was implemented in JavaScript. Today there are several other implementations in different languages available. To show case how to use GraphQL, we will explore the `graphql-python` implementation in this chapter. The official documentation of GraphQL is available at [\[117\]](#)

## 21.2.2 Prerequisites

Before we start we need to install a number of tools that we use throughout the chapter

### 21.2.2.1 Install Graphene

In this chapter, we will use [Graphene](#) which is a library for implementing GraphQL APIs in Python. Use `pip` to install Graphene

```
$ pip install graphene==2.0.1 graphene-django==2.0.0
```

### 21.2.2.2 Install Django

For the purpose of demonstrating in this chapter, we will use Django as Python web framework. Django is a popular Python web framework which already comes with a lot of boilerplate code. It is mature and has a very large community. It has inbuilt support for Object Relational Mapping which is based on Database Code-First approach. Please refer [118] for more Django information. Use `pip` to install Django

```
$ pip install django==2.0.2 django-filter==1.1.0
```

### 21.2.2.3 Install GraphiQL

In case you prefer to use a browser interface which could be useful for debugging purposes a number of GraphQL browsers are available. A free version is GraphiQL. It is an IDE(Interactive Development Environment) for GraphQL where you can run a GraphQL Query. There are many implementations of GraphiQL available. For this chapter we will use [GraphiQL](#). You can download the GraphiQL installation files specific to your OS from [GraphiQL Releases](#).

For MacOS, you can even use `homebrew` to install it

```
brew cask install graphiql
```

Commercial GraphQL browsers are available from

- [Insomnia](#)
- [Altair](#)

## 21.2.3 GraphQL type system and schema

To get started with GraphQL we will first explore the GraphQL type system and schema creation.

### 21.2.3.1 Type System

In GraphQL a query is what a client requests from the GraphQL

server. The result will be obtained in a structure defined by type and schema. Thus, the client will know ahead of time what it is going to get as result as part of a well formed response. For this to work, the data is often assumed to be structured data.

To demonstrate the type system we will use a simple example while looking at authors and co-authors of papers. We represent in this example a database that contains a number of authors. Each author has a publication count and a number of coauthors that are identified by name. We assume for this simple example that all author names are unique.

Here is how a simple GraphQL query would look like

```
{
 author {
 name
 publication_count
 coauthors {
 name
 }
 }
}
```

The response is

```
{
 "author": {
 "name": "John Doe",
 "publication_count": 25,
 "coauthors": [
 {
 "name": "Mary Jane"
 },
 {
 "name": "David Miller"
 }
]
 }
}
```

For this to work, we need to define the types that are going to be honored by the GraphQL service so that when a query is received by the server, it is first validated against a schema that defines the types contained within the GraphQL service.

Hence, types must be defined as part of each GraphQL service. They

are defined with the GraphQL schema language which is programming language agnostic. An example of a GraphQL type is:

```
type Author {
 name: String!
 publication_count: Int
 coauthors: [Author!]!
}
```

Here we define the type `author` with three fields `name`, `publication`, and `coauthors`. Note that the `!` indicates a field value, that cannot be null and must have a defined value. `[Author!]!` means that an array is returned, but that array cannot be null and also none of the items in the array can be null.

### 21.2.3.2 Scalar Types

GraphQL supports the following scalar types:

- `String`: UTF8 characters
- `Int`: 32 bit signed integer
- `Float`: Double precision floating point value
- `Boolean`: true or false
- `ID`: Represents a unique identifier which can be used as a key to fetch the object

### 21.2.3.3 Enumeration Types

`enum` is a scalar type which defines restricted set of values. When a GraphQL schema defines a field of `enum` type, we expect that the field's value be of the type `enum` including only the values that are included in that enumeration. An example of an `enum` type is

```
enum ContainerType {
 Docker
 Kubernetes
 DockerSwarm
}
```

### 21.2.3.4 Interfaces

Similar to common programming languages, the GraphQL type system also supports an `interface`. Interfaces allow us to assure that certain fields are part of the definition of a type. When a type implements an `interface`, it needs to specify all the fields that are defined through the `interface`.

We will illustrate this in the following example, where we define simple `computeService` interface type. This `interface` declares `id`, `name` and `memory` fields. This means that a `Container` and a `VirtualMachine` both of which implement `ComputeService`, must have the fields defined in the `interface`. They may or may not have additional fields like we demonstrate in our example with the field `systemType` of type `ContainerType` in case of `Container` and field `systemType` of type `VMBackend` in case of the `VirtualMachine`.

```
interface ComputeService {
 id: ID!
 name: String!
 memory: Int!
}

type Container implements ComputeService {
 id: ID!
 name: String!
 memory: Int!
 systemType: ContainerType!
}

type VirtualMachine implements ComputeService {
 id: ID!
 name: String!
 memory: Int!
 systemType: VMBackend!
}
```

### 21.2.3.5 Union Types

As the name suggests a `union` type represents the union of two or more types. The following example shows how we can define a `union` type. As you can see we use the `|` character to indicate the union operator.

```
union ComputeType = Container | VirtualMachine
```

Now when we write a GraphQL query to get the `computeType` information, we can ask some of the common fields and some of the specific fields conditionally. In the next example we request `AllComputeTypes` with

common fields like `id`, `name` and fields specific to either `VirtualMachine` or `Container`.

```
{
 AllComputeTypes {
 id
 name
 ... on VirtualMachine {
 user
 }
 ... on Container {
 type
 }
 }
}
```

## 21.2.4 GraphQL Query

An application asks for data from server in form of a GraphQL query. A GraphQL query can have different fields and arguments and in this section we describe how to use them.

### 21.2.4.1 Fields

A very simple definition of a query is to ask for specific fields that belong to an object stored in GraphQL.

In the next examples we will use data related to repositories in github.

When asking the query

```
{
 repository {
 name
 }
}
```

we obtain the following response

```
{
 "data": {
 "repository": {
 "name": "cm"
 }
 }
}
```

As we can see the response data format looks exactly like the query. This way a client knows exactly what data it has to consume. In the previous example, the `name` field returns the data of type `String`. Clients can also ask for an object representing any match within the GraphQL database.

For example following query

```
{
 community {
 name
 repositories {
 name
 }
 }
}
```

returns the response

```
{
 "data": {
 "community": {
 "name": "cloudmesh-community",
 "repositories": [
 {"name": "S.T.A.R boat"},
 {"name": "book"}
]
 }
 }
}
```

#### 21.2.4.2 Arguments

As you may already know in REST services you can pass parameters as part of a request via query parameters through GET or a request body through POST. However in GraphQL, for every field, you provide an argument restricting the data returned to only the information that you need. This reduces the traffic for returning the information that is needed without doing the postprocessing on the client. These restricting arguments can be of scalar type, enumeration type and others.

Let us look at an example of a query where we only ask for first 3 repositories in cloudmesh community

```
{
 repositories(first: 3) {
 name
 url
 }
}
```

The response will be similar to

```
{
 "data": {
 "repositories": [
 {
 "name": "boat",
 "url": "https://github.com/cloudmesh-community/boat"
 }, {
 "name": "book",
 "url": "https://github.com/cloudmesh-community/book"
 }, {
 "name": "case",
 "url": "https://github.com/cloudmesh-community/case"
 }
]
 }
}
```

### 21.2.4.3 Fragments

Fragments allow us to reuse portions of a query. Let us look at the following complex query, which includes repetitive fields:

```
{
 boatRepositoryExample: repository(name: boat) {
 name
 full_name
 url
 description
 }
 cloudRepositoryExample: repository(name: cm) {
 name
 full_name
 url
 description
 }
}
```

As the query gets bigger and more complex, we can use a `fragment` to split it into smaller chunks. This `fragment` can then be re-used, which can significantly reduce the query size and also make it more readable.

A `fragment` can be defined as

```
fragment repositoryInfo on Repository {
 name
 full_name
```

```
 url
 description
}
```

and can be used in a query like this

```
{
 boatRepositoryExample: repository(name: boat) {
 ...repositoryInfo
 }
 cloudRepositoryExample: repository(name: cm) {
 ...repositoryInfo
 }
}
```

The response for this query will look like

```
{
 "data": {
 "boatRepositoryExample": {
 "name": "boat",
 "fullName": "cloudmesh-community/boat",
 "url": "https://github.com/cloudmesh-community/boat",
 "description": "S.T.A.R. boat"
 },
 "cloudRepositoryExample": {
 "name": "cm",
 "fullName": "cloudmesh-community/cm",
 "url": "https://github.com/cloudmesh-community/cm",
 "description": "Cloudmesh v4"
 }
 }
}
```

#### 21.2.4.4 Variables

Variables are used to pass dynamic values to queries. Instead of passing hard-coded values to a query, variables can be defined for these values. Now these variables can be passed to queries.

Variables can be passed to GraphQL queries directly through commandline. Please note that we pretty print the json output with python's `json.tool`. So it is not actually part of the querry, but convenient to format the output. Try to see the difference with and without the pipe to `json.tool`

```
curl -X POST \
-H "Content-Type: application/json;" \
-d '{ "query": "{ repository (name: $name) { name url } }", "variables": \
{ "name": "book" } }' \
| json.tool
```

```
http://localhost:8000/graphq1/ | python -m json.tool
```

In case you use GraphQL, variables can be defined in the Query Variables panel at left bottom of the GraphQL client. It is defined as a JSON object and this is how it looks like

```
{
 "name": "book"
}
```

and it can be used in the query like this

```
{
 repository(name: $name) {
 name
 url
 }
}
```

which will result in the response

```
{
 "data": {
 "repository": {
 "name": "book",
 "url": "https://github.com/cloudmesh-community/book"
 }
 }
}
```

#### 21.2.4.5 Directives

Directives are used to change the structure of queries at runtime using variables. Directives provide a way to describe additional options to GraphQL executors. Currently the core GraphQL specification supports two directives

- `@skip (if: Boolean)` - It skips the field if argument is true
- `@Include (if: Boolean)` - It includes the field if argument is true

To demonstrate its usage, we define the variable `isAdmin` and assign a value of `true` to it.

```
{
 "isAdmin": true
}
```

This variable is passed as an argument `showOwnerInfo` to the query. This argument is in turn passed to `@include` directive to determine whether to include the `ownerInfo` sub-query.

```
{
 repositories(showOwnerInfo: $isAdmin) {
 name
 url
 ownerInfo @Include(if: $showOwnerInfo) {
 name
 }
 }
}
```

Since we have defined `showOwnerInfo` as `true`, the response includes `ownerInfo` data.

```
{
 "data": {
 "repositories": [{
 "name": "book",
 "url": "https://github.com/cloudmesh-community/book",
 "ownerInfo": {
 "name": "cloudmesh-community"
 }
 }]
 }
}
```

#### 21.2.4.6 Mutations

Mutations are used to modify the server side data. To demonstrate this, let us look at the query and the data to be passed along with it

```
mutation CreateRepositoryForCommunity($community: Community!, $repository: Repository!) {
 createRepository(community: $community, repository: $repository) {
 name
 url
 }
}

{
 "community": "cloudmesh-community",
 "repository": {
 "name": "cm-burn",
 "url": "https://github.com/cloudmesh-community/cm-burn"
 }
}
```

The response will be as follow, indicating that a repository has been added.

```
{
 "data": {
 "createRepository": {
 "name": "cm-burn",
 "url": "https://github.com/cloudmesh-community/cm-burn"
 }
 }
}
```

## 21.2.4.7 Query Validation

GraphQL is a language with strong type system. So requesting and providing wrong data will generate an error.

For example the query

```
{
 repositories {
 name
 url
 type
 }
}
```

will give the response

```
{
 "errors": [{
 "message": "Cannot query field \"type\" on type \"Repository\".",
 "locations": [
 {"line": 5,
 "column": 3}
]
 }]
}
```

In an application we need to validate the user input. If it is invalid we can use the `GraphQLError` class or Python exceptions to raise validation errors.

Let us take an example of mutation query. We want to validate whether repository name is empty or not. We can use `GraphQLError` to raise validation error from our mutation function like this

```
def mutate(self, info, url, name, full_name, description):
 if not name:
 raise GraphQLError('Repository name is required')
 repository = Repository(url=url, name=name, full_name=full_name, description=description)
 repository.save()
```



## 21.2.5 GraphQL in Python

We will cover a basic server implementation with schema and queries to fetch and mutate data.

To develop a GraphQL server in Python we will use `Django` as Python web framework and the `Graphene` library which allows us to develop GraphQL APIs. Naturally other frameworks such as Flask could be used, but for this example we focus on Django. The installation for Graphene and Django been already described at the beginning of this chapter. Our example is located in the book repository which you can clone with

```
$ git clone https://github.com/cloudmesh-community/book.git
```

The example itself is located in the directory

- [book/examples/graphql/cloudmeshrepo](#) - A graphql server example with local database

To execute the example you need to go in the specific directory. Thus, for `cloudmeshrepo` say

```
$ cd book/examples/graphql/cloudmeshrepo
```

Then you can execute following steps

```
$ pip install django-graphql-jwt==0.1.5
$ python manage.py migrate
$ python manage.py runserver
```

The last command will start a server on localhost and you can access it at

- <http://localhost:8000>

It will show you a graphical interface based on GraphiQL, allowing you to execute your queries in it.

## 21.2.6 Developing your own GraphQL Server

If you want to create GraphQL server while using django as the web server backend yourself, you can start with following steps

```
$ mkdir -p example/graphql
$ cd example/graphql
$ pip install django-graphql-jwt==0.1.5
$ django-admin startproject cloudmeshrepo
$ cd cloudmeshrepo
$ python manage.py migrate
$ python manage.py runserver
```

The last command will start a server on the localhost and you can access it at the URL

- <http://localhost:8000>

It will show you the welcome page for django. Now open the file

cloudmeshrepo/cloudmeshrepo/settings.py

file under folder and append following to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
 # After the default packages
 'graphene_django',
)
```

At the end of `settings.py` add following line

```
GRAPHENE = {
 'SCHEMA': 'cloudmeshrepo.schema.schema',
}
```

### 21.2.6.1 GraphQL server implementation

Clients can request for data to GraphQL server via GraphQL queries. They can also use mutations to insert data into GraphQL server's database. Django follows the principle of separating different modules in a project into apps. For this example, we will have two apps, one for Users and one for Repositories. For the demo purpose, we have decided not use backend such as MongoDB but instead we will use SQLite.

Django provides `startapp` utility to create blank app with some

biolerplate code.

Go to the root dir of project and execute the following command which will create an app for repository.

```
python manage.py startapp repository
```

Now open `Repositories/models.py` and add the `Repository` model class.

```
class Repository(models.Model):
 url = models.URLField()
 name = models.TextField(blank=False)
 full_name = models.TextField(blank=False)
 description = models.TextField(blank=True)
```

Now open the file `cloudmeshRepository/settings.py` and append following line into INSTALLED\_APPS

```
INSTALLED_APPS = (
 # After the graphene_django app
 'Repositories',
)
```

Go to root folder and execute following commands. It will create table for new model

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Naturally, for us to demonstrate the server, we need to ingest some data into the server. We can easily do this with the django shell while calling

```
$ python manage.py shell
```

Inside the shell, execute following command to create some example data. We have taken this data from github's API and used the repositories in the cloudmesh community at

- <https://api.github.com/users/cloudmesh-community/repos>.

You could use either `wget` or `curl` command to download this data through shell. As this data is huge, we have used a small subset for this example. You can have a python script, shell script or any other

program to clean and remodel the data as per your need; the implementation details for the cleaning process is out of scope for this chapter.

```
from Repositories.models import Repository
Repository.objects.create(
 name="boat",
 full_name="cloudmesh-community/boat",
 url="https://github.com/cloudmesh-community/boat",
 description="S.T.A.R. boat")
Repository.objects.create(
 name="book",full_name="cloudmesh-community/book",
 url="https://github.com/cloudmesh-community/book",
 description="Gregor von Laszewski")
Repository.objects.create(name="cm",
 full_name="cloudmesh-community/cm",
 url="https://github.com/cloudmesh-community/cm",
 description="Cloudmesh v4")
Repository.objects.create(name="cm-burn",
 full_name="cloudmesh-community/cm-burn",
 url="https://github.com/cloudmesh-community/cm-burn",
 description="Burns many SD cards so we can build a Raspberry PI cluster")
exit()
```

Now create the file `Repositories/schema.py` with the following code. The code will create a custom type `Repository` and query with a resolver function for `Repositories`. The GraphQL server uses a resolver function to resolve the incoming queries. Queries can respond to only those fields or entities of schema for which resolver function has been defined. A `Resolver` function's responsibility is to return data for that specific field or entity. We will create one for `Repositories` list. When you query `repositories`, resolver function will return all the `repositories` objects from database.

```
import graphene
from graphene_django import DjangoObjectType

from .models import Repository

class RepositoryType(DjangoObjectType):
 class Meta:
 model = Repository

class Query(graphene.ObjectType):
 Repositories = graphene.List(RepositoryType)

 def resolve.Repositories(self, info, **kwargs):
 return Repository.objects.all()
```

Next create the file `cloudmeshRepository/schema.py` with following code. It just inherits the query defined in Repositories app. This way we are able to isolate schema to their apps.

```
import graphene

import Repositories.schema

class Query(Repositories.schema.Query, graphene.ObjectType):
 pass

schema = graphene.Schema(query=Query)
```

### 21.2.6.2 GraphQL Server Querying

Next, we create a Schema and use it within GraphiQL which is a playground for GraphQL queries. Open the file `cloudmeshrepository/urls.py` and append the following code

```
from django.views.decorators.csrf import csrf_exempt
from graphene_django.views import GraphQLView

urlpatterns = [
 path('admin/', admin.site.urls),
 path('graphql/', csrf_exempt(GraphQLView.as_view(graphiql=True))),
]
```

Start your server using the command

```
$ python manage.py runserver
```

Now open in your browser the URL

- <http://localhost:8000/graphql>

You will see GraphiQL window. In the left pane you can add queries. Let us add the following query

```
{
 repositories {
 name
 fullName
 url
 description
 }
}
```

```
}
```

In the right pane you will see following output

```
{
 "data": {
 "repositories": [
 {
 "name": "boat",
 "fullName": "cloudmesh-community/boat",
 "url": "https://github.com/cloudmesh-community/boat",
 "description": "S.T.A.R. boat"
 },
 {
 "name": "book",
 "fullName": "cloudmesh-community/book",
 "url": "https://github.com/cloudmesh-community/book",
 "description": "Gregor von Laszewski"
 },
 {
 "name": "cm",
 "fullName": "cloudmesh-community/cm",
 "url": "https://github.com/cloudmesh-community/cm",
 "description": "Cloudmesh v4"
 },
 {
 "name": "cm-burn",
 "fullName": "cloudmesh-community/cm-burn",
 "url": "https://github.com/cloudmesh-community/cm-burn",
 "description": "Burns many SD cards so we can build a Raspberry PI cluster"
 }
]
 }
}
```

### 21.2.6.3 Mutation example

Similar to a query, you can add a mutation to create your own data. To achieve this, add a `createRepository` class for new repository object which will inherit from graphene's Mutation class. This class will accept a new repository as an argument. Please see the following code snippet which is added to `repositories/models.py`.

```
class CreateRepository(graphene.Mutation):
 url = graphene.String()
 name = graphene.String()
 full_name = graphene.String()
 description = graphene.String()

 class Arguments:
 url = graphene.String()
 name = graphene.String()
 full_name = graphene.String()
```

```

 description = graphene.String()

 def mutate(self, info, url, name, full_name, description):
 repository = Repository(url=url, name=name,
 full_name=full_name,
 description=description)
 repository.save()

 return CreateRepository(url=repository.url,
 name=repository.name,
 full_name=repository.full_name,
 description=repository.description)

```

Similar to A Query, add a `Mutation` class in the repository's schema in `repositories/schema.py`.

```

class Mutation(graphene.ObjectType):
 create_repository = CreateRepository.Field()

```

Now you can run the following mutation on GraphiQL to add a new repository

```

mutation {
 createRepository (
 url: "https://github.com/cloudmesh-community/repository-test",
 name: "repository-test",
 fullName: "cloudmesh-community/repository-test",
 description: "Test repository"
) {
 url
 name
 fullName
 description
 }
}

```

This will insert a new repository `repository-test` and also immediately return its inserted data fields (`url, name, fullName, description`).

```

{
 "data": {
 "createRepository": {
 "url": "https://github.com/cloudmesh-community/repository-test",
 "name": "repository-test",
 "fullName": "cloudmesh-community/repository-test",
 "description": "Test repository"
 }
 }
}

```

## 21.2.6.4 GraphQL Authentication

There are a number of ways to add authentication to your GraphQL server

- We can add a REST API endpoint which will take care of authenticating the user and only the logged in users can make GraphQL queries. This method can also be used to restrict only a subset of GraphQL queries. This is ideal for existing applications, which have REST endpoints, and which are trying to migrate over to GraphQL.
- We can add basic authentication to the GraphQL server which will just accept credentials in raw format and once authenticated, logged in user can start GraphQL querying
- We can add JSON Web Token authentication to GraphQL server, since most of the applications these days are stateless.

#### **21.2.6.5 JSON Web Token Authentication**

Next we focus on the JSON web token (JWT) authentication. It is typically preferred as it provides a more secure and sophisticated way of authentication. As part of the authentication process, a client has to provide a username and a password. A limited life time token is generated that is used during the authentication process. Once the token is generated, it needs to be provided with each subsequent GraphQL API call to assure the authentication is valid.

JWT tokens are bearer tokens which need to be passed in HTTP authorization header. JWT tokens are very safe against CSRF attacks and are trusted and verified since they are digitally signed.

The advantage of using frameworks such as the python implementation of GraphQL is that it can leverage existing authentication modules, so we do not have to develop them ourselves. One such module is JSON Web Token Authentication or \*JWT Authentication. To use this module, please add it to the `settings.py` file as follows

```
MIDDLEWARE = [
```

```
'graphql_jwt.middleware.JSONWebTokenMiddleware',
[

AUTHENTICATION_BACKENDS = [
 'graphql_jwt.backends.JSONWebTokenBackend',
 'django.contrib.auth.backends.ModelBackend',
]
```

Add the Token mutation to `cloudmeshrepo/schema.py`.

```
class Mutation(users.schema.Mutation, repositories.schema.Mutation, graphene.ObjectType):
 token_auth = graphql_jwt.ObtainJSONWebToken.Field()
```

Run the server using `runserver` command and fire the token mutation providing username and password. You can either run this mutation on GraphQL or using `curl` command.

For GraphQL run this on query panel.

```
mutation {
 tokenAuth (username:"user1", password:"Testing123") {
 token
 }
}
```

Or if you are on bash shell, use this `curl` command

```
curl -X POST \
-H "Content-Type: application/json;" \
-d '{"query": "{ mutation { tokenAuth (username: \"user1\", ' \
'password:\"Testing123\") { token } } }"}' \
http://localhost:8000/graphql/ | python -m json.tool
```

This will create a token for us to use in our subsequent calls.

```
{
 "data": {
 "tokenAuth": {
 "token": "eyJ0eXAiOiJKV1.... (cut to fit in line)"
 }
 }
}
```

The JWT library comes with a built-in directive called `login_required`. You can add this to any of your Query resolvers to prevent unauthenticated access. We have annotated it to the `resolve_repositories` which means it will throw authentication error to query which does not have JWT token passed. Whenever a valid JWT token is present in

the query, it is considered as authenticated or logged in request, and data will be served only to these queries.

```
from graphql_jwt.decorators import login_required
...

class Query(graphene.ObjectType):
 repositories = graphene.List(RepositoryType)

 @login_required
 def resolve_repositories(self, info, **kwargs):
 return Repository.objects.all()
```

Now if you try to query our repositories from GraphQL, you will see this error

```
{
 "errors": [
 {
 "message": "You do not have permission to perform this action",
 "locations": [
 {
 "line": 2,
 "column": 3
 }
],
 "path": [
 "repositories"
]
 },
 "data": {
 "repositories": null
 }
}
```

Henceforth you need to pass token with every repository query. This token needs to be passed as header. Unfortunately, the GraphiQL UI client does not support this. Hence you can use either a curl query from command line or more advanced GraphQL clients that support authentication.

#### 21.2.6.5.1 Using Authentication with Curl

To use authentication with curl, you can pass the token to the command. For simplicity we created a TOKEN environment variable in which we store the token so it is easier for us to refer to it in our examples.

```

export TOKEN=eyJ0eXAiOiJKV1.... (cut to fit in line)
curl -X POST \
-H "Content-Type: application/json;" \
-H "Authorization: JWT $TOKEN" \
-d '{"query": "{ repositories { url } }"}' \
http://localhost:8000/graphql/ | python -m json.tool

```

The result obtained from running this command is:

```

{"data": {"repositories": [
 {"url": "https://github.com/cloudmesh-community/boat"},
 {"url": "https://github.com/cloudmesh-community/book"},
 {"url": "https://github.com/cloudmesh-community/cm"},
 {"url": "https://github.com/cloudmesh-community/cm-burn"},
 {"url": "https://github.com/cloudmesh-community/vineet-test-1"},
 {"url": "https://github.com/cloudmesh-community/vineet-test"}
]}
}

```

#### 21.2.6.5.2 Expiration of JWT tokens

JWT tokens have a time-to-life and expire after a while. This is controlled by the GraphQL server and is usually communicated to the client in transparent documented fashion.

If the token is about to expire, you can call the `refreshToken` mutation to refresh the Token and to return the refreshed token to the client. However, if the token has already expired we will need to request a new token by calling `tokenAuth` mutation.

More information about JWT tokens can be found at [\[119\]](#) and the GraphQL authentication page at [\[120\]](#).

#### 21.2.6.6 GitHub API v4

GraphQL has made already an impact in the cloud services community. In addition to Facebook, Twitter and Pinterest, Github is now also providing a GraphQL interface, making it an ideal example for us.

GitHub has implemented as part of its API v4 also GraphQL which allows you to query or mutate data of repositories that you can access via [github.com](https://github.com). To demonstrate its use, we will use GraphiQL.

To access the information, we need an OAuth token to access GitHub API. You can generate an OAuth token by following the steps listed at

- <https://help.github.com/articles/creating-a-personal-access-token-for-the-command-line/>

Next we demonstrate the use of Github within a GraphQL browser called Open GraphiQL. First you need to click edit headers at upper-right corner and add a new header with key Authorization and value Bearer your\_token.

Next you enter the URL

- <https://api.github.com/graphql>

in the GraphQL endpoint textbox and keep the method as POST only. To test if the changes have been applied successfully you can use the query

```
query {
 viewer {
 login
 name
 }
}
```

The query gives the following response

```
{
 "data": {
 "viewer": {
 "login": "*Your GitHub UserId*",
 "name": "*Your Full Name*"
 }
 }
}
```

To get a list of our own repositories add following query

```
query($number_of_repositories:Int!) {
 viewer {
 name
 repositories(last: $number_of_repositories) {
 nodes {
 name
 }
 }
 }
}
```

```
}
```

To limit the responses we can define a use the variable `number_of_repositories`

```
{
 "number_of_repositories": 3
}
```

The query gives the following response

```
{
 "data": {
 "viewer": {
 "name": "*your name*",
 "repositories": {
 "nodes": [
 {
 "name": "*Repository 1*"
 },
 {
 "name": "*Repository 2*"
 },
 {
 "name": "*Repository 3*"
 }
]
 }
 }
 }
}
```

To add a comment using mutation we need to get the issue `id` with the query

```
{
 repository(owner:"MihirNS", name:"Temp_Repository") {
 issue(number: 1) {
 id
 }
 }
}
```

The query gives the following response

```
{
 "data": {
 "repository": {
 "issue": {
 "id": "MDU6SXNzdWUzNjUxMDIwOTE="
 }
 }
 }
}
```

Now we can use the id as `subjectId` for mutation to add a comment to an issue with the query

```
mutation AddComment {
 addComment(input:{subjectId:"MDU6SXNzdWUzNjUxMDIwOTE=", body:"This comment is done using
GitHub API v4"}) {
 commentEdge {
 node {
 repository{
 nameWithOwner
 }
 url
 }
 }
 }
}
```

The query gives the following response

```
{
 "data": {
 "addComment": {
 "commentEdge": {
 "node": {
 "repository": {
 "nameWithOwner": "MihirNS(Temp_Repository"
 },
 "url": "https://github.com/MihirNS(Temp_Repository/issues/1#issuecomment-4256203"
 }
 }
 }
 }
}
```

## 21.2.7 Dynamic Queries with GraphQL

The previous examples served data to and from a database. However, often we need to access dynamic data that is provided through function or system calls.

For this reason we like you to be reminded about the Section describing the [resolver function](#). It allows us to add functions on the server side that return the data from various data sources.

It is similar in nature than our example in the REST OpenAPI section, where we associate call backs that execute dynamic operations. More information about the functionality in REST is provided in the Section [OpenAPI Specification](#) as part of the [path definition](#).

## 21.2.8 Advantages of Using GraphQL

Unlike REST APIs, only the required data is fetched minimizing the data transferred over network.

Separation of concern is achieved between client and server. Client requests data entities with fields needed for the UI in one query request while server knows about the data structure and how to resolve the data from its sources which could be database, web service, microservice, external APIs, etc.

Versioning is simpler than REST, since we only have to take care of it when we want to remove any of the fields. We can even introduce the property of a deprecated field for a while to inform the service users. At a later time the field could be entirely removed.

```
type Car {
 id: ID!
 make: String
 description: String @deprecated(reason: "Field is deprecated!")
}
```

## 21.2.9 Disadvantages of Using GraphQL

GraphQL query can get very complex. A client may not necessarily know how expensive the queries can be for server to go and gather the data. This can be overcome by limiting, for example, the query depth and recursion.

Caching gets pretty tricky and messy in case of GraphQL. In REST, you can have separate API url for each resource requested, caching can be done at this resource level. However, in GraphQL you can have different queries but they can operate over a single API url. This means that caching needs to be done at the field level introducing additional complexity.

## 21.2.10 Conclusion

GraphQL is gaining momentum as growing and the integration into services such as Github, Pinterest, Intuit, Coursera, and Shopify,

demonstrates this. Many GraphQL editors, IDEs and packages have recently been developed.

In general there are several reasons to use GraphQL due to its simplicity and flexibility. It also fits well with the microservices architecture which is a new trend in cloud architectures. With that being said, REST APIs still have their own place and may prove better choice in certain use cases. Both REST and GraphQL have some tradeoffs which need to be understood before making a choice between the one or the other. Github shows that both can be used.

### 21.2.10.1 Resources

- Official documentation of Github API v4 is available at [\[121\]](#)
- More GraphQL Python examples available at [\[122\]](#)

### 21.2.11 Exercises

#### E.GraphQL.1:

The chapter shows you how to develop a GraphQL server with Django. Instead of Django we like you to use Flask. Develop a documented section showcasing this. Use the resolver class to demonstrate a dynamic information example such as the cpu type. Showcase integration with mongoengine and dynamic information retrieval from another information source.

See an example

- [https://github.com/graphql-python/graphene-mongo/tree/master/examples/flask\\_mongoengine](https://github.com/graphql-python/graphene-mongo/tree/master/examples/flask_mongoengine)
- <https://graphene-mongo.readthedocs.io/en/latest/tutorial.html>

#### E.GraphQL.2:

Develop a GraphQL server and client that queries your

CPU information through a dynamic query using a resolver

#### E.GraphQL.3: OpenStack VMS

Develop a GraphQL server that returns the information of running virtual machines on OpenStack

#### E.GraphQL.4: OpenStack Azure

Develop a GraphQL server that returns the information of running virtual machines on OpenStack

#### E.GraphQL.5: OpenStack Aws

Develop a GraphQL server that returns the information of running virtual machines on OpenStack

#### E.GraphQL.6: Cloud Service

Pick a Cloud Service and develop a GraphQL interface for it.

#### E.GraphQL.7: Cloudmesh

Develop a cloudmesh framework that uses all clouds above while returning the information of all running VMS in a Web page. You are allowed to make the Web page beautiful with HTML5 and/or JavaScript if you have the background to do so. Contact Gregor if you like to work on this for your project.

## **21.3 PYTHON APACHE AVRO**



Although Apache Avro is not directly a messaging system, it uses messaging to communicate between components while serializing and deserializing objects defined with a schema. In addition it provides data structures, remote procedure call (RPC), a container file to store

persistent data and simple integration with dynamic languages [123]. Avro depends on schemas, which are defined with JSON. This facilitates implementation in other languages that have the JSON libraries. The key advantages of Avro are schema evolution - Avro will handle the missing/extra/modified fields, dynamic typing - serialization and deserialization without code generation, untagged data - data encoding and faster data processing by allowing data to be written without overhead.

The following steps illustrate using Avro to serialize and deserialize data with example modified from Apache Avro 1.8.2 Getting Started (Python) [124].

### 21.3.1 Download, Unzip and Install

The zipped installation file avro-1.8.2.tar.gz could be downloaded from [here](#)

To unzip, using linux:

```
tar xvf avro-1.8.2.tar.gz
```

using MacOS:

```
gunzip -c avro-1.8.2.tar.gz | tar xopf -
```

cd into the directory and install using the following:

```
cd avro-1.8.2
python setup.py install
```

To check successful installation, import avro in python without error message:

```
python
>>> import avro
```

This instruction is for Python2. The Python3 counterpart, avro-python3-1.8.2.tar.gz could be downloaded from [here](#) and the unzip and install procedure is the same.

## 21.3.2 Defining a schema

Use a simple schema for students contributed in cloudmesh as an example: paste the following lines into an empty text file with the name student.avsc

```
{"namespace": "cloudmesh.avro",
"type": "record",
"name": "Student",
"fields": [
 {"name": "name", "type": "string"},
 {"name": "hid", "type": "string"},
 {"name": "age", "type": ["int", "null"]},
 {"name": "project_name", "type": ["string", "null"]}
]
}
```

This schema defines a record representing a hypothetical student, which is defined to be a record with the name Student and 4 fields, namely name, hid, age and project name. The type of each of the field needs to be provided. If any field is optional, one could use the list including null to define the type as shown in age and project name in the example schema. Further, a namespace cloudmesh.avro is also defined, which together with the name attribute defines the full name of the schema (cloudmesh.avro.Student in this case).

## 21.3.3 Serializing

The following piece of python code illustrates serialization of some data

```
import avro.schema
from avro.datafile import DataFileWriter
from avro.io import DatumWriter

schema = avro.schema.parse(open("student.avsc", "rb").read())

writer = DataFileWriter(open("students.avro", "wb"), DatumWriter(), schema)
writer.append({"name": "Albert Zweistein",
 "hid": "hid-sp18-405",
 "age": 99,
 "project_name": "hadoop with docker"})
writer.append({"name": "Ben Smith",
 "hid": "hid-sp18-309",
 "project_name": "spark with docker"})
writer.append({"name": "Alice Johnson",
 "hid": "hid-sp18-208",
```

```
"age": 27})
writer.close()
```

The code does the following:

- Imports required modules
- Reads the schema student.avsc (make sure that the schema file is placed in the same directory as the python code)
- Create a DataFileWriter called writer, for writing serialized items to a data file on disk
- Use DataFileWriter.append() to add data points to the data file. Avro records are represented as Python dicts.
- The resulting data file saved on the disk is named students.avro
- This above instruction is for Python2. If one is using Python3, change

```
schema = avro.schema.parse(open("student.avsc", "rb").read())
```

to:

```
schema = avro.schema.Parse(open("student.avsc", "rb").read())
```

since the method name has a different case in Python3.

### 21.3.4 Deserializing

The following python code illustrates deserialization

```
from avro.datafile import DataFileReader
from avro.io import DatumReader

reader = DataFileReader(
 open("students.avro", "rb"), DatumReader())
for student in reader:
 print (student)
reader.close()
```

The code does the following:

- Imports required modules
- Use DatafileReader to read the serilaized data file

- students.avro, it is an iterator
- Returns the data in a python dict

The output should look like:

```
{'name': 'Albert Zweistein',
 'hid': 'hid-sp18-405',
 'age': 29,
 'project_name': 'hadoop with docker'
}
{'name': 'Ben Smith',
 'hid': 'hid-sp18-309',
 'age': None,
 'project_name': 'spark with docker'
}
{'name': 'Alice Johnson',
 'hid': 'hid-sp18-208',
 'age': 27,
 'project_name': None
}
```

### 21.3.5 Resources

- The steps and instructions are modified from [Apache Avro 1.8.2 Getting Started \(Python\) \[124\]](#).
- The Avro Python library does not support code generation, while Avro used with Java supports code generation, see [Apache Avro 1.8.2 Getting Started \(Java\) \[125\]](#).
- Avro provides a convenient way to represent complex data structures within a Hadoop MapReduce job. Details about Avro are documented in [Apache Avro 1.8.2 Hadoop MapReduce guide \[126\]](#).
- For more information on schema files and how to specify name and type of a record can be found at [record specification \[127\]](#).

## 21.4 APCHE FLINK O ?



O student can contribute a chapter

## 22 FAQ



### 22.1 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.

#### 22.1.1 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.

#### 22.1.2 Piazza

##### 22.1.2.1 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.

### **22.1.3 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.

### **22.1.4 Has SOIC computers I can use remotely?**

See: <https://uisapp2.iu.edu/confluence-prd/pages/viewpage.action?pageId=114491559>

### **22.1.5 When contributing to the book my name is not listed properly or not at all**

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

## **22.1.6 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 juste 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.**

## 22.1.7 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\\_yml.py](https://github.com/cloudmesh-community/book/tree/master/examples/yaml-validation/validate_yml.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_yml.py
$./validate_yml.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>

## 22.1.8 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

## 22.1.9 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.yml` 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.

## 22.1.10 How to open the epub from Github?

If you see thw `view Raw`, you need to click on it. It will download the file. Than 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.

## 22.1.11 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

## 22.1.12 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
```

## **22.1.13 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?

### **22.1.14 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 your 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.

## **22.2 FAQ: 516**



This section contains FAQs relevant for 516. faq

### **22.2.1 Why have the individual 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 in 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 with 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.

### **22.2.2 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 put 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.

### **22.2.3 Microsoft Edge for Epub**

On Windows 10 you need to have version 42.17134 for edge to properly display an epub.

## 22.2.4 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 correctly.

## 22.2.5 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](https://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's 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 finds 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.

## **22.2.6 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.

## **22.2.7 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 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
```

## **22.2.8 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

### **22.2.9 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 your 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 your 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.



### 23.1 AMAZON WEB SERVICE PRODUCTS



#### 23.1.1 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](#)

#### 23.1.2 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](#)

#### 23.1.3 Databases

- [Amazon Aurora](#)
- [Amazon RDS](#)
- [Amazon DynamoDB](#)
- [Amazon ElastiCache](#)
- [Amazon Redshift](#)
- [Amazon Neptune](#)
- [AWS Database Migration Service](#)
- [AWS Marketplace](#)

### 23.1.4 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](#)

### 23.1.5 Networking & Content Delivery

- [Amazon VPC](#)
- [Amazon VPC PrivateLink](#)
- [Amazon CloudFront](#)
- [Amazon Route 53](#)
- [Amazon API Gateway](#)
- [AWS Direct Connect](#)
- [Elastic Load Balancing](#)
- [AWS Marketplace](#)

### 23.1.6 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](#)

### 23.1.7 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](#)

### 23.1.8 Media Services

- [Amazon Elastic Transcoder](#)
- [Amazon Kinesis Video Streams](#)
- [AWS Elemental MediaConvert](#)
- [AWS Elemental MediaLive](#)
- [AWS Elemental MediaPackage](#)
- [AWS Elemental MediaStore](#)
- [AWS Elemental MediaTailor](#)

### 23.1.9 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](#)

### 23.1.10 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](#)

### 23.1.11 Analytics

- [Amazon Athena](#)
- [Amazon EMR](#)

- [Amazon CloudSearch](#)
- [Amazon Elasticsearch Service](#)
- [Amazon Kinesis](#)
- [Amazon Redshift](#)
- [Amazon QuickSight](#)
- [AWS Data Pipeline](#)
- [AWS Glue](#)
- [AWS Marketplace](#)

### 23.1.12 Mobile

- [AWS Mobile Hub](#)
- [Amazon API Gateway](#)
- [Amazon Pinpoint](#)
- [AWS AppSync](#)
- [AWS Device Farm](#)
- [AWS Mobile SDK](#)

### 23.1.13 AR & VR

- [Amazon Sumerian](#)

### 23.1.14 Application Integration

- [Amazon MQ](#)
- [Amazon Simple Queue Service \(SQS\)](#)
- [Amazon Simple Notification Service \(SNS\)](#)
- [AWS AppSync](#)
- [AWS Step Functions](#)

### 23.1.15 Customer Engagement

- [Amazon Connect](#)
- [Amazon Pinpoint](#)
- [Amazon Simple Email Service \(SES\)](#)
- [AWS Marketplace](#)

## **23.1.16 Business Productivity**

- [Alexa for Business](#)
- [Amazon WorkDocs](#)
- [Amazon WorkMail](#)
- [Amazon Chime](#)

## **23.1.17 Desktop & App Streaming**

- [Amazon WorkSpaces](#)
- [Amazon AppStream 2.0](#)

## **23.1.18 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](#)

## **23.1.19 Game Development**

- [Amazon GameLift](#)
- [Amazon Lumberyard](#)

## **23.1.20 AWS Marketplace Software**

- [Infrastructure Software \(1300+\)](#)
- [Business Software \(845+\)](#)
- [Developer Tools \(220+\)](#)

## **23.1.21 AWS Cost Management**

- [AWS Cost Explorer](#)

- [AWS Budgets](#)
- [Reserved Instance Reporting](#)
- [AWS Cost and Usage Report](#)

### **23.1.22 Exercise**

E.AWS.0:

Identify all products related to IaaS service offerings and mark them with ★ 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.

## **23.2 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 onece markd with a ★ are related to IaaS.

The services are organized in teh 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.

### **23.2.1 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](#)

## 23.2.2 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](#)

## 23.2.3 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](#)

### **23.2.4 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](#)

### **23.2.5 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](#)

## 23.2.6 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](#)

### 23.2.7 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](#)

## 23.2.8 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](#)

## 23.2.9 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](#)

## 23.2.10 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](#)

## 23.2.11 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](#)

### 23.2.12 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](#)

### 23.2.13 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.

### 23.2.14 Migration

Source: <https://support.microsoft.com/en-us/allproducts>

[Migration](https://azure.microsoft.com/en-us/services/azure-migrate/)(<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”)

### 23.2.15 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](#)

## 23.2.16 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](#)

## 23.2.17 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](#)

### **23.2.18 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-us/services/#storage> “See more”)

## 23.2.19 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



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.

### 24.1 FutureSystems Evolved from FutureGrid

---

In this video we introduce FutureGrid a precursor to FutureSystems.

[Systems 12:12 FutureGrid](#)

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.

### 24.2 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](#)

### 24.3 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](#)

## **24.4 SHELL ACCESS VIA SSH**

---

This lesson explains how to get access FutureSystems resources via SSH terminal with your registered SSH key.

 [Python 2:34 Shell Access via SSH](#)

## **24.5 ADVANCED SSH**

---

This lesson shows you how to write SSH 'config' file in advanced settings.

 [Python 2:47 Advanced SSH](#)

## **24.6 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 your FutureSystems portal, and then login using the PuTTY SSH client.

 [Python 3:51 Windows users](#)

## **24.7 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.

### **24.7.1 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.

### **24.7.2 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

### **24.7.3 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

## **24.7.4 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

## **24.7.5 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

## **24.7.6 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

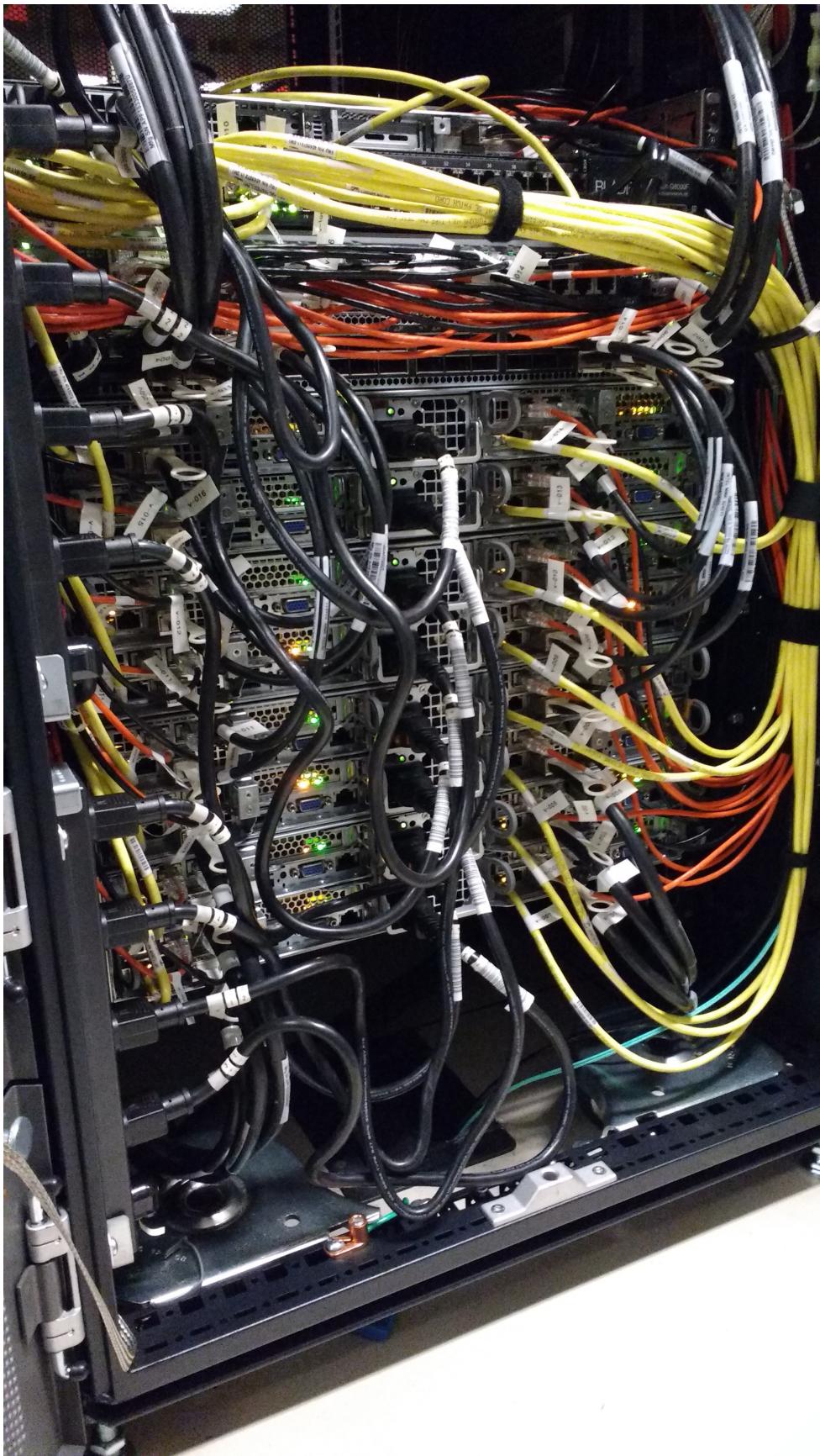
7250F 1.50GHz processor. All nodes with 200GB of memory, 3.2TB of local disk storage, 800GB of NVMe storage, and an Intel OmniPath adapter for high bandwidth, low-latency MPI applications. Operating System: CentOS release 7.2.1511

### **24.7.7 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

### **24.7.8 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 202](#))



923

Figure 202: Cabling of Victor

## **24.7.9 PI Cluster**

The details will be added once we have completed the purchase.



### 25.1 CHAMELEON CLOUD SECURITY WARNING



**⚠ Chameleon cloud promotes insecure use of ssh while suggesting passphrase less keys.** This is **very dangerous** due to the fact that someone could gain access to your computer and if a password less key is stolen easy access to other systems can be achieved. Instead you must use whenever possible passphrases and use ssh agent and ssh add!

To show you this insecure practice, we quote from their [Web page](#):

You will receive a message “Enter same passphrase again:” so just leave it blank and press enter.

Hence do not use their advise that is mentioned multiple times in their documentation. Follow ours, and use a passphrase!

### 25.2 RESOURCES

You can also visit the Chameleon Web page as there is also more information about other topics that we may not need to worry about. Furthermore we do not use Advanced Appliances, but use mostly ansible instead as it is independent from OpenStack and can be used with other frameworks.

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. We would like to get your feedback in case you find errors or like to contribute to this documentation.

The links to Chameleons online resources are:

- [Web page](#)

- [Documentation](#)
- [News](#)
- [About](#)
- [Log in](#)
- [Dashborad](#)

### 25.2.1 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/>

To be notified by mail, you can subscribe to them at

<https://www.chameleoncloud.org/user/profile/subscriptions/>

### 25.2.2 Account Creation

The fist 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 als consult the chameleon chapter in the handbook.

### 25.2.3 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-18 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 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.

## 25.2.4 Usage Restriction

As using VM's in a shared environment cost resources, 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.

## 25.3 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    |

[Detailed information in Chameleon cloud Resource Discovery Portal](#)

### 25.3.1 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.



Chameleon Cloud Racks

### 25.3.2 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.

### 25.3.3 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.

### **25.3.4 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 K80  | GK 210 | 2496 × 2      | 24 GiB 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.

### 25.3.5 Live updates

You can browse detailed information about the resources offered for bare metal reconfiguration in the [Resource Discovery Portal](#).

## 25.4 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.

#### **25.4.1 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  $\frac{[(10 + 1) \text{ SUs} \text{ per hour}]}{[7 \text{ days} \cdot 24 \text{ hours/day}]} = 2,016 \text{ SUs}$  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.

## 25.4.2 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.

## **25.5 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:

### **25.5.1 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.

## **25.5.2 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 chameleoen cloud administrators. As they do not like taht because you have not followed the chameleon cloud project policies you will receive a grade deduction for the class.

### **25.5.3 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 below:

[\*\*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.

## **25.6 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.

### **25.6.1 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 203](#)

The initial log in page appears as:



Figure 203: Chameleon login

After a successful log in, you will see the Overview page as shown below. 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 below. 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 below, the quota and usage apply to the current project you have selected and no information about your other projects is shown. See [Figure 204](#)

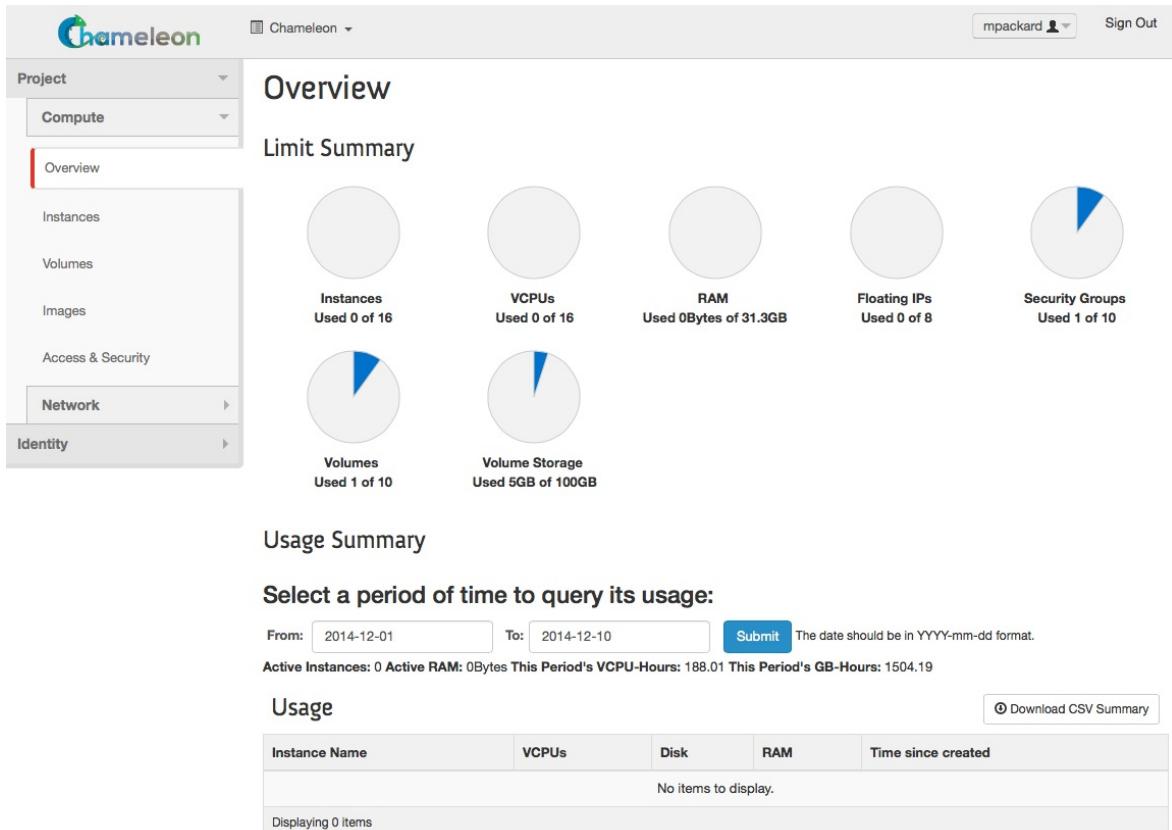


Figure 204: Overview page

### 25.6.1.1 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 below. 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 205](#)

The screenshot shows the Chameleon Instances page. On the left, there's a sidebar with 'Project' dropdown set to 'Compute', 'Overview', 'Instances' (which is selected), 'Volumes', 'Images', 'Access & Security', 'Network', and 'Identity'. The main area is titled 'Instances' with a sub-section 'Instances'. It has a toolbar with 'Instance Name' dropdown, 'Filter', 'Launch Instance' (highlighted in blue), 'Soft Reboot Instances', and 'Terminate Instances'. A table lists two instances: 'Ubuntu instance' (Ubuntu-Server-14.04-LTS) and 'CentOS image' (CentOS-7). The table columns include Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions (with a 'Create Snapshot' button). At the bottom, it says 'Displaying 2 items'.

Figure 205: 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 206](#)

**Launch Instance**

**Details \***    **Access & Security \***    **Networking \***    **Post-Creation \***    **Advanced Options**

|                                                                         |                                                                                               |
|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| <b>Availability Zone</b>                                                | Specify the details for launching an instance.                                                |
| Alamo                                                                   | The chart below shows the resources used by this project in relation to the project's quotas. |
| <b>Instance Name *</b>                                                  | CentOS image                                                                                  |
| <b>Flavor *</b> ?                                                       | <b>m1.small</b>                                                                               |
| Some flavors not meeting minimum image requirements have been disabled. |                                                                                               |
| <b>Instance Count *</b> ?                                               | 1                                                                                             |
| <b>Instance Boot Source *</b> ?                                         | Boot from image                                                                               |
| <b>Image Name</b>                                                       | CentOS-7 (329.2 MB)                                                                           |
| <b>Flavor Details</b>                                                   |                                                                                               |
| Name                                                                    | m1.small                                                                                      |
| VCPUs                                                                   | 1                                                                                             |
| Root Disk                                                               | 20 GB                                                                                         |
| Ephemeral Disk                                                          | 0 GB                                                                                          |
| Total Disk                                                              | 20 GB                                                                                         |
| RAM                                                                     | 2,048 MB                                                                                      |
| <b>Project Limits</b>                                                   |                                                                                               |
| Number of Instances                                                     | 2 of 16 Used                                                                                  |
| Number of VCPUs                                                         | 2 of 16 Used                                                                                  |
| Total RAM                                                               | 4,096 of 32,000 MB Used                                                                       |
| <b>Cancel</b> <b>Launch</b>                                             |                                                                                               |

Figure 206: 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 207](#)

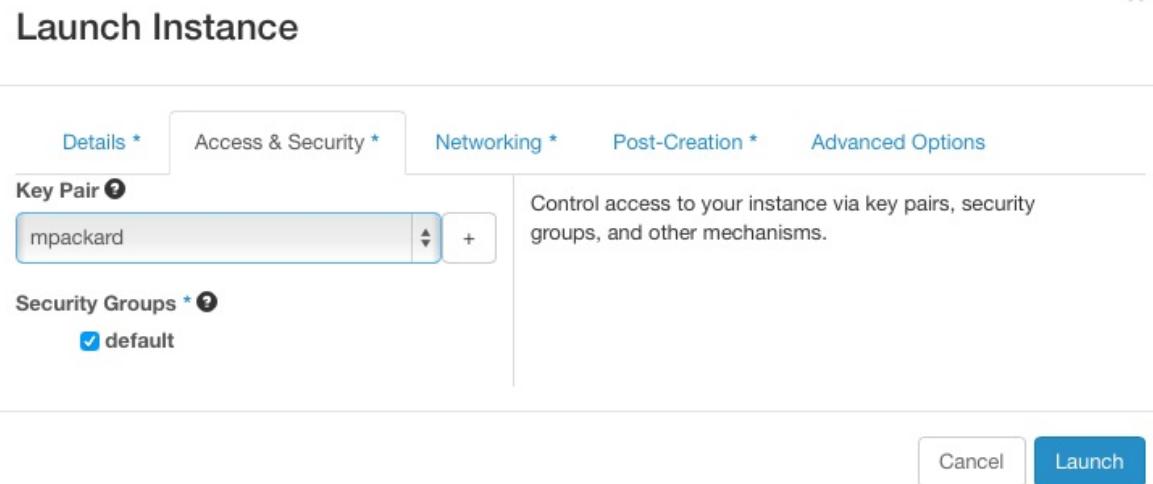


Figure 207: Access window

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 208](#)

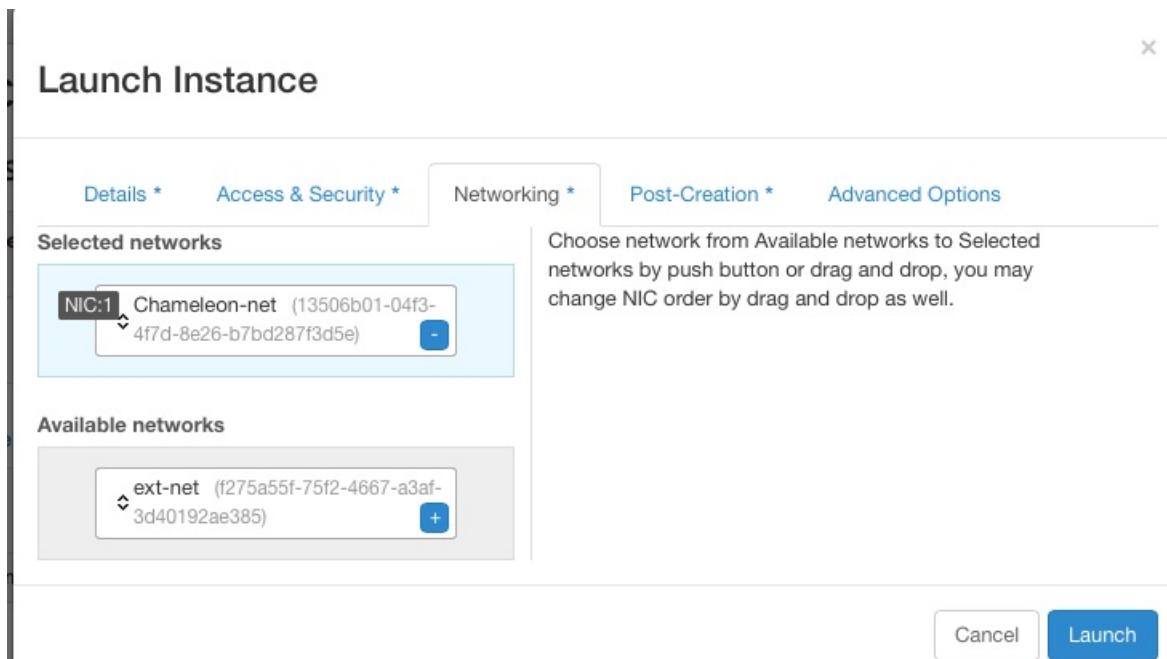


Figure 208: Networking window

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 209](#)

The screenshot shows a user interface for managing floating IP associations. At the top, a header reads "Manage Floating IP Associations". Below the header, there is a form with two main sections. The first section is labeled "IP Address \*" and contains a dropdown menu with the value "129.114.32.32". To the right of this dropdown is a note: "Select the IP address you wish to associate with the selected instance." The second section is labeled "Port to be associated \*" and contains a dropdown menu with the value "test: 192.168.0.57". At the bottom right of the form are two buttons: "Cancel" and "Associate".

Figure 209: 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 210](#)



Figure 210: Console

### 25.6.1.2 Snapshots

The instance list page shown above 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.

### 25.6.1.3 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 211](#)

## Access & Security

The screenshot shows a web-based interface for managing security groups. At the top, there are tabs for 'Security Groups', 'Key Pairs', 'Floating IPs', and 'API Access'. The 'Security Groups' tab is selected. Below the tabs is a header bar with 'Create Security Group' and 'Delete Security Groups' buttons. The main area is titled 'Security Groups' and contains a table with one item. The table has columns for 'Name' (with a checkbox), 'Description', and 'Actions'. The item listed is 'default' with 'default' in the description and a 'Manage Rules' button in the actions column. A message at the bottom says 'Displaying 1 item'.

Figure 211: 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 212](#)

### Manage Security Group Rules: default

The screenshot shows a 'Manage Security Group Rules' page for the 'default' group. It features a header with 'Add Rule' and 'Delete Rules' buttons. Below is a table listing six rules. The columns are 'Direction', 'Ether Type', 'IP Protocol', 'Port Range', 'Remote', and 'Actions'. Each rule includes a checkbox and a 'Delete Rule' button. The rules are: 1. Ingress, IPv4, Any, -, default. 2. Egress, IPv6, Any, -, ::/0 (CIDR). 3. Egress, IPv4, Any, -, 0.0.0.0/0 (CIDR). 4. Ingress, IPv6, Any, -, default. 5. Ingress, IPv4, ICMP, -, 0.0.0.0/0 (CIDR). 6. Ingress, IPv4, TCP, 22 (SSH), 0.0.0.0/0 (CIDR). A message at the bottom says 'Displaying 6 items'.

Figure 212: 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 213](#)

## Add Rule

The screenshot shows the 'Add Rule' configuration page. On the left, there are three dropdown fields: 'Rule \*' set to 'SSH', 'Remote \*' set to 'CIDR', and 'CIDR' set to '0.0.0.0/0'. To the right, a 'Description:' section provides information about security group rules. It states that rules define traffic allowed to instances assigned to the security group, consisting of three parts: rule template, source, and target. It details the 'Rule' field for specifying TCP, UDP, or ICMP rules, the 'Open Port/Port Range' field for defining port ranges, and the 'Remote' field for specifying the source of traffic via IP address block or security group. At the bottom right are 'Cancel' and 'Add' buttons.

**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 213: Add a security group

### 25.6.2 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.

### 25.6.3 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” section above, 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

## 25.7 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 to the 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.

## 25.7.1 OpenStack RC File

We will use the Nova command line tools for Chameleon OpenStack and to authorize our account on the command line tools. To do so, you will need an openstack RC file.

### 25.7.1.1 Creating OpenStack RC via the editor

The easiest way is to create this file by hand while copying the following lines into the file `~/.cloudmesh/chameleon/cc-openrc.sh`. Make sure that you place the file in a location you easily be found:

```
$ mkdir -p ~/.cloudmesh/chameleon
```

The easiest way is to download a template from pur book with

```
$ wget https://raw.githubusercontent.com/cloudmesh/book/master/examples/chameleon/cc-openrc.sh
```

The 'cc-openrc.sh' looks as follows:

```
#!/bin/bash

export CC_PROJECTID="CH-819337"
export CC_PREFIX="albert-111" # repalce with your username and hid number

export OS_AUTH_URL=https://openstack.tacc.chameleoncloud.org:5000/v2.0
With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT

export OS_TENANT_ID=$CC_PROJECTID
export OS_TENANT_NAME=$CC_PROJECTID
export OS_PROJECT_NAME=$CC_PROJECTID
export OS_USERNAME=<put your chameleon cloud username here>

export OS_REGION_NAME="RegionOne"
if [-z "$OS_REGION_NAME"]; then unset OS_REGION_NAME; fi
```

Please make sur to replace `<put your chameleon cloud username here>` with your chameleon cloud username. Now whenever you need top access chameleon cloud you can use the command

```
$ source ~/.cloudmesh/chameleon/cc-openrc.sh
```

To simplify the configuration and documentation, we have included two shell environment variables. The first one is `cc_PROJECT`, that specifies the project number. The second one is a prefix that you will use for VMS and keys as we are using a shared project. This way we can see which VMS and which keys have been uploaded and keep the names of them unique.

```
$ export CC_PROJECT=CH-819337
$ export CC_PREFIX=albert-111
```

### 25.7.1.2 Creating OpenStack RC via the GUI

In case you do not want to use the commandline option to obtain an RC sample, you can obtain the OpenStack RC file with the OpenStack Dashboard.

<https://openstack.tacc.chameleoncloud.org/dashboard>

Login and chose your project number for this project. Confirm your project number and find **Access & Security** on the left menu. The Access & Security page has tabs and choose **API Access** to download credentials on a local machine. Click **Download OpenStack RC File** to download CH-\$PROJECTID-openrc.sh file on your machine (see [Figure 214](#)). Every time you use nova command line tools, the file should be loaded on your terminal.

| Service   | Service Endpoint                                              |
|-----------|---------------------------------------------------------------|
| Compute   | https://openstack.tacc.chameleoncloud.org:8774/v2/CH-819337   |
| Network   | https://openstack.tacc.chameleoncloud.org:9696                |
| Volumev2  | https://openstack.tacc.chameleoncloud.org:8776/v2/CH-819337   |
| Computev3 | https://openstack.tacc.chameleoncloud.org:8774/v3             |
| Image     | https://openstack.tacc.chameleoncloud.org:9292                |
| Volume    | https://openstack.tacc.chameleoncloud.org:8776/v1/CH-819337   |
| EC2       | https://openstack.tacc.chameleoncloud.org:8773/services/Cloud |
| Identity  | https://openstack.tacc.chameleoncloud.org:5000/v2.0           |

Figure 214: Access and Security GUI

```
$ mkdir -p ~/.cloudmesh/chameleon
$ mv ~/Downloads/CH-$CC_PROJECT-openrc.sh ~/.cloudmesh/chameleon/cc-openrc.sh
```

Just as in the previous section please add the following to your openrc.sh file while adapting it appropriately.

```
$ export CC_PROJECT=CH-819337
$ export CC_PREFIX=albert-111
```

Once you source the file, you can use nova command line tools without sourcing it again. The environment variables are enabled while your terminal is alive. In case you have not stored the original RC file in the Downloads folder, please copy it from that location instead.

## 25.7.2 CLI to Manage Virtual Machines

OpenStack provides a commandline tool called nova to manage virtual machines. To install it please use the command

```
$ pip install python-openstackclient
```

To see if your configuration works and the command is installed, make sure you have the `cc-openrc.sh` file and sourced it. Than issue the

command

```
$ nova image-list
```

You will see an output similar to

| ID                               | Name            | Status | Server |
|----------------------------------|-----------------|--------|--------|
| be46bd5a-c4a5-4495-ad30-35618... | CC-C7-autologin | ACTIVE |        |
| 1fe5138b-300b-4b30-8d22-e7287... | CC-CentOS7      | ACTIVE |        |
| ...                              |                 |        |        |

### 25.7.3 Creating SSH keys

Naturally you will need an ssh key. If you do not have an existing SSH keypair, you can create one. Please see Section ??[C:ssh]?? for more details:

```
$ ssh-keygen -t rsa -C albert@example.edu
```

### 25.7.4 KeyPair Registration

Once you have completed the installation of nova, you also need to register a ssh keypair with openstack to be able to log into the virtual machines that you start. To register your public key, use:

```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub $CC_PREFIX-key
```

Once you register your key, you can confirm if your key registration has been successful by listing the keys:

```
$ nova keypair-list
```

You will see an output similar to:

| Name            | Fingerprint                                     |
|-----------------|-------------------------------------------------|
| \$CC_PREFIX-key | cf:04:06:aa:8b:76:af:77:aa:0a:b5:87:ff:0f:ba:97 |

### 25.7.5 Start a new VM instance

To start new instances you can use the nova boot command. It will start a VM instance. You can use some parameters to specify which base image and a server size we will use with a name. We use CC-Ubuntu16.04 base image in this section which is an official Ubuntu 16.04 image provided by Chameleon project.

```
$ nova boot --image CC-Ubuntu16.04 --key-name $CC_PREFIX-key --flavor m1.small $CC_PREFIX-01
```

where the 01 indicates the instance number. Note that we will be terminating and deleting any VM in our project that does not follow this naming convention.

## 25.7.6 Floating IP Address

If your new VM instance is up and running, it needs an external ip address which is also called floating IP address. A floating IP allows you to get access to this VM from the internet. Note that chameleon has a limited number of floating IP addresses and it is best to return them if not in use. If chameleon runs out of floating IP addresses, please submit a ticket to chameleon. However in many cases the VM may only need a an internal IP address as a default. In case you need to access others, you could even tunnel all connections through a single floating IP. naturally this would limit data transfers in and out of chameleon, but is a recommended way to deal with limited floating IPs.

Let us showcase how to associate a floating IP address and access it via SSH.

```
nova floating-ip-create ext-net
+-----+-----+-----+-----+-----+
| Id | IP | Server Id | Fixed IP | Pool |
+-----+-----+-----+-----+-----+
| 13dc309e-... | 129.114.111.37 | - | - | ext-net |
+-----+-----+-----+-----+
```

Now we have a IP address to assign to a VM instance. In this section, we will associate 129.114.111.37 to our albert-111-01 VM instance by:

```
$ nova floating-ip-associate albert-111-01 129.114.111.37
```

Once you completed this step, you are now able to SSH into your VM instance. Confirm ACTIVE state in your VM to get access.

|          |               |        |   |         |                                 |
|----------|---------------|--------|---|---------|---------------------------------|
| f19e1... | albert-111-01 | ACTIVE | - | Running | \$CC_PROJECT-net=               |
|          |               |        |   |         | 192.168.0.13,<br>129.114.111.37 |

where 111 is the number from your hid and 01 is the instance number

```
$ ssh cc@129.114.111.37
```

Note that cc is login name your VM if you start a VM with the official Chameleon cloud image.

## 25.7.7 Termination of VM Instance

If you completed your work on your VM instance, you have to terminate your VM and release a floating IP address associated with. For example, we terminate our first instance and the IP address by:

```
$ nova delete $CC_PREFIX-01
$ nova floating-ip-delete 129.114.111.37
```

Please note that when using delete you will delete the VM. In case you still need to use it use `stop` and to restart it use `start` instead.

## 25.8 OPENSTACK HORIZON GRAPHICAL USER INTERFACE

### 25.8.1 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 215](#)

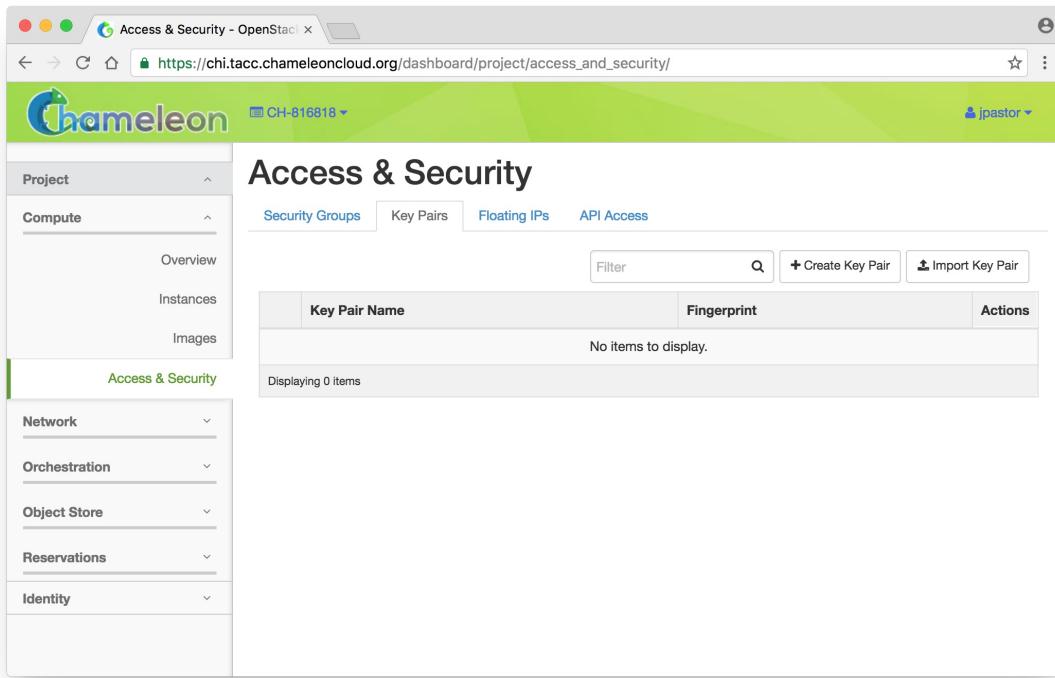


Figure 215: Key Pairs Tab

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 216](#)

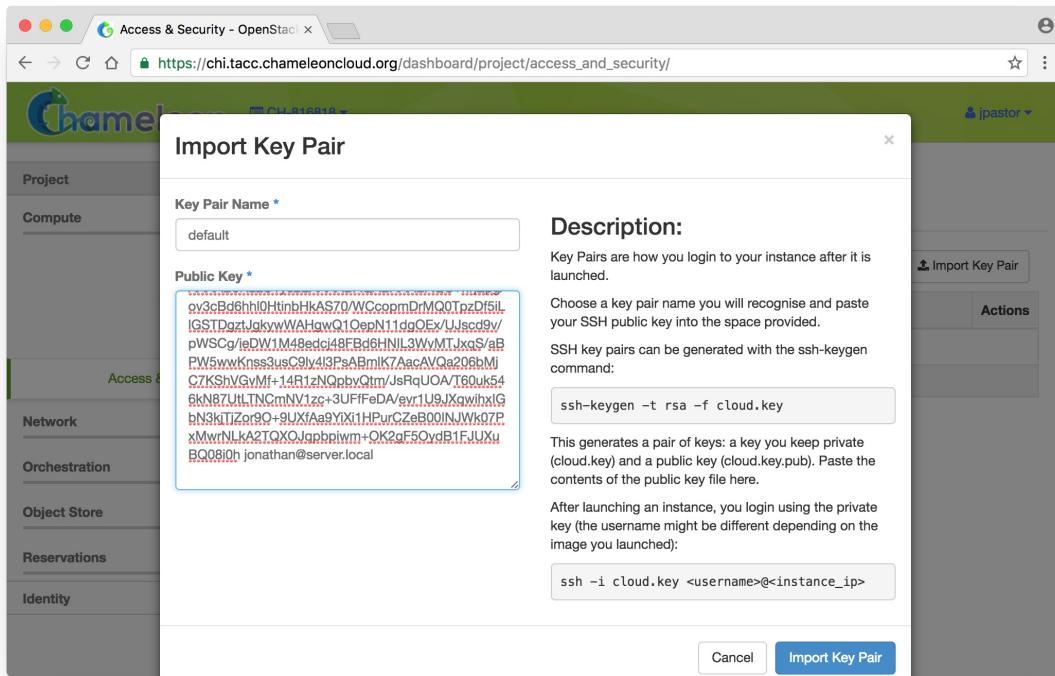


Figure 216: Public Key

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 217](#)

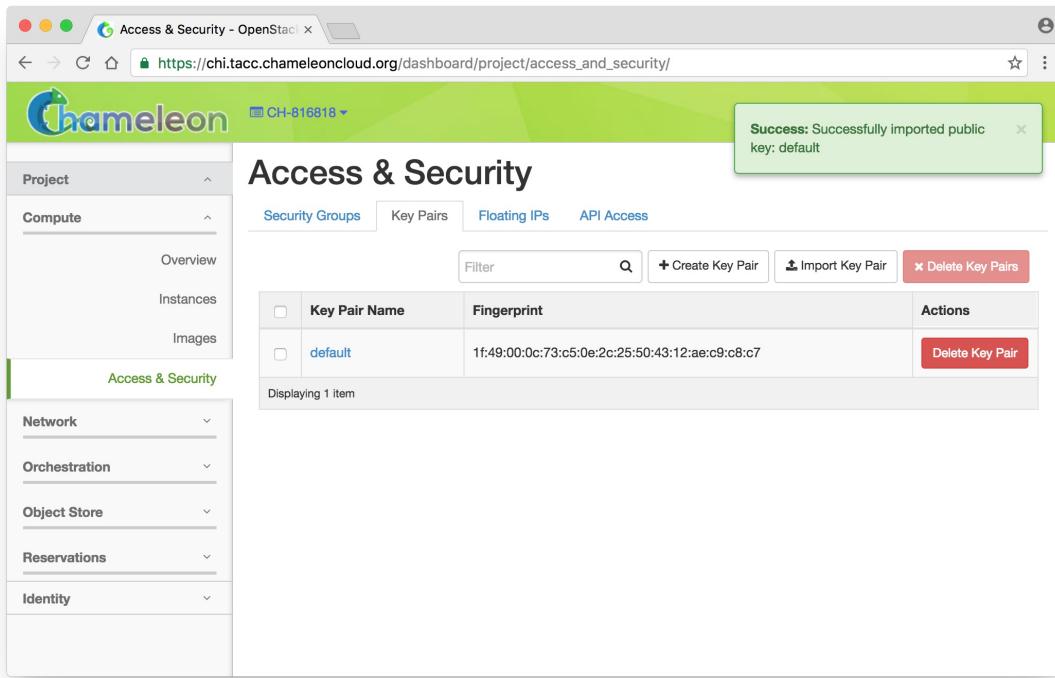


Figure 217: Import key pair

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 218](#)

The screenshot shows the Chameleon OpenStack Dashboard interface. The left sidebar has a 'Compute' section with 'Instances' selected. The main area is titled 'Instances' and displays a table with one row: 'No items to display.' Below the table, it says 'Displaying 0 items'. At the top right of the main area, there is a 'Launch Instance' button.

Figure 218: 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 219](#)

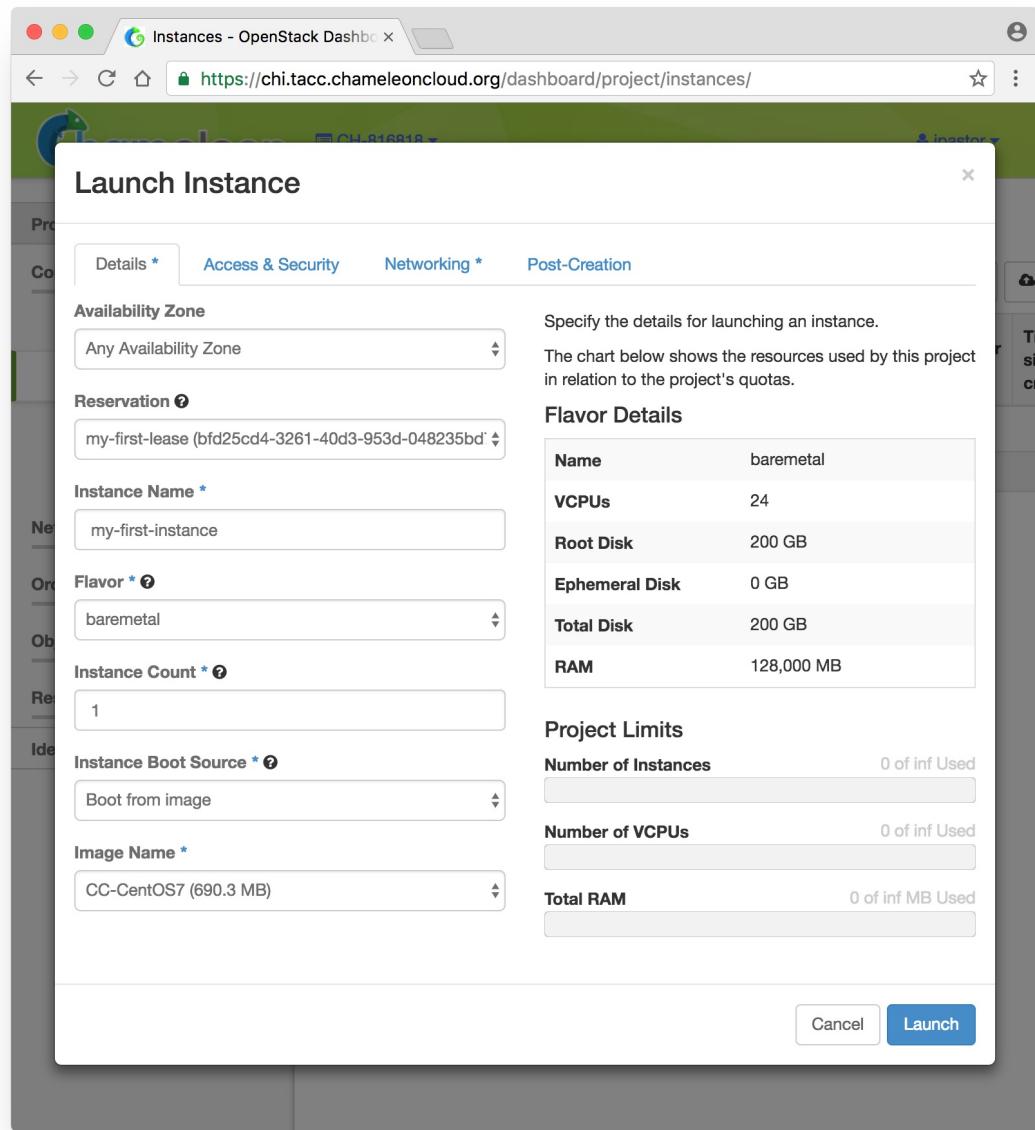


Figure 219: Launch a VM

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 220](#)

Success: Launched instance named "my-first-instance".

|                          | Instance Name     | Image Name | IP Address | Size      | Key Pair | Status | Availability Zone | Task       | Power State | Time since created | Actions                               |
|--------------------------|-------------------|------------|------------|-----------|----------|--------|-------------------|------------|-------------|--------------------|---------------------------------------|
| <input type="checkbox"/> | my-first-instance | CC-CentOS7 |            | baremetal | default  | Build  |                   | Scheduling | No State    | 0 minutes          | <a href="#">Associate Floating IP</a> |

Figure 220: Status Window (a)

After a few minutes the instance should become in Active status and the Power State should be Running. See [Figure 221](#)

|                          | Instance Name     | Image Name | IP Address  | Size      | Key Pair | Status | Availability Zone                            | Task | Power State | Time since created | Actions                                |
|--------------------------|-------------------|------------|-------------|-----------|----------|--------|----------------------------------------------|------|-------------|--------------------|----------------------------------------|
| <input type="checkbox"/> | my-first-instance | CC-CentOS7 | 10.40.0.164 | baremetal | default  | Active | climate:fd0b6542-4851-4d2e-aa11-0441732cecc0 | None | Running     | 4 minutes          | <button>Associate Floating IP</button> |

Figure 221: 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 the one below: See [Figure 222](#)

Figure 222: 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 223](#)

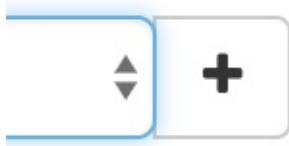


Figure 223: Allocate the IP

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 224](#)

The screenshot shows a modal dialog box titled "Manage Floating IP Associations". Inside the dialog, there are two input fields: "IP Address \*" and "Port to be associated \*". The "IP Address \*" field contains the value "129.114.108.90". The "Port to be associated \*" field contains the value "my-first-instance: 10.40.0.164". To the right of the "IP Address" field is a note: "Select the IP address you wish to associate with the selected instance or port." At the bottom right of the dialog are two buttons: "Cancel" and "Associate", with "Associate" being highlighted in blue.

Figure 224: 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 225](#)

| Instance Name     | Image Name | IP Address                                  | Size      | Key Pair | Status | Availability Zone                            | Task | Power State | Time since created | Actions                                  |
|-------------------|------------|---------------------------------------------|-----------|----------|--------|----------------------------------------------|------|-------------|--------------------|------------------------------------------|
| my-first-instance | CC-CentOS7 | 10.40.0.164<br>Floating IPs: 129.114.108.90 | baremetal | default  | Active | climate:fd0b6542-4851-4d2e-aa11-0441732cecc0 | None | Running     | 31 minutes         | <a href="#">Disassociate Floating IP</a> |

Figure 225: Status of the IP Association

## 25.8.2 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 226](#)

```
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 226: 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.

### 25.8.2.1 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](#).

### 25.8.3 Use FPGAs

Consult the [dedicated page](#) if you would like to use the FPGAs available on Chameleon.

### 25.8.4 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](#)

## 25.9 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).

### 25.9.1 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 above 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](#).

## 25.9.2 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 227](#).

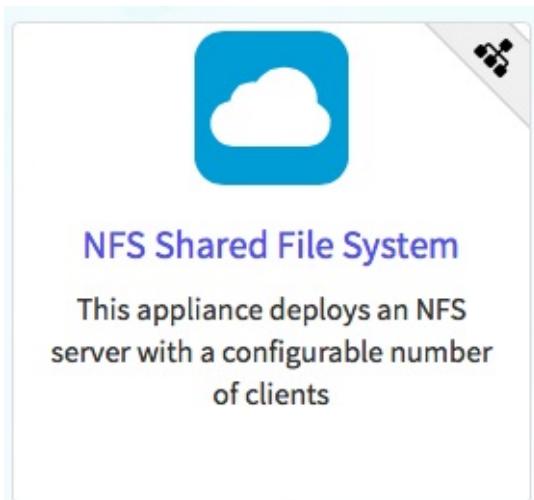


Figure 227: NFS file share

## 25.9.3 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 below: [Figure 228](#).

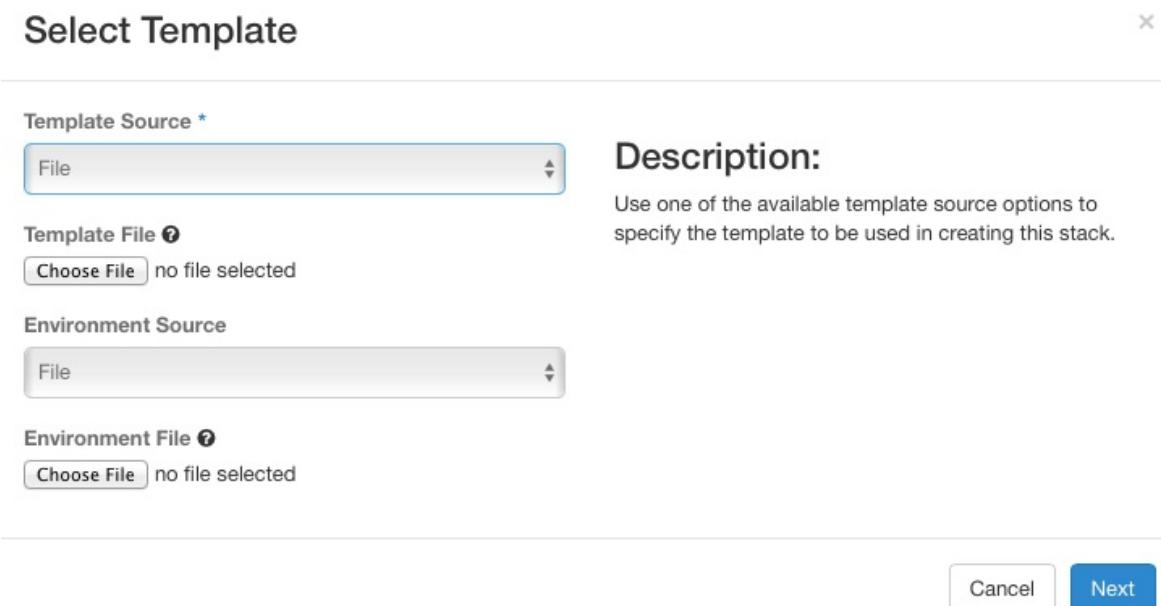


Figure 228: Select Template

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 you 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 229](#).

**Launch Stack**

---

**Stack Name \*** ?

**Description:**  
Create a new stack with the provided values.

**Creation Timeout (minutes) \*** ?

 ↑ ↓

Rollback On Failure ?

**Password for user "priteau" \*** ?

 eye

**key\_name** ?

 ↑ ↓

**nfs\_client\_count** ?

 ↑ ↓

**reservation\_id \*** ?

 ↑ ↓

---

Figure 229: 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 below: [Figure 230](#).

|                          | Stack Name     | Created   | Updated | Status             | Actions                        |
|--------------------------|----------------|-----------|---------|--------------------|--------------------------------|
| <input type="checkbox"/> | my-nfs-cluster | 0 minutes | Never   | Create In Progress | <button>Check Stack</button> ▾ |

Figure 230: Create in Progress

It will then move to the status “Create Complete”. As the following: [Figure 231](#).

|                          | Stack Name     | Created    | Updated | Status          | Actions                        |
|--------------------------|----------------|------------|---------|-----------------|--------------------------------|
| <input type="checkbox"/> | my-nfs-cluster | 15 minutes | Never   | Create Complete | <button>Check Stack</button> ▾ |

Figure 231: Create Complete

You can click on the stack name to get more details, including a visualization of the deployed resources, as pictured below: [Figure 232](#). 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!

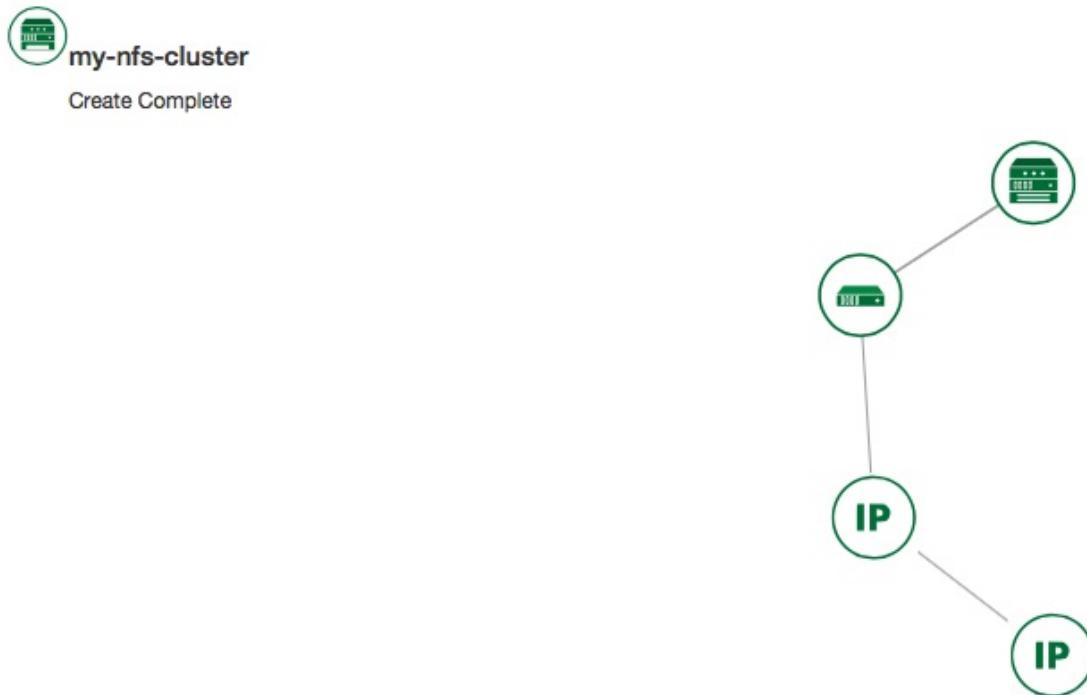


Figure 232: Stack Visualization

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 above) 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 above (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.

## 25.9.4 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 below, 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] }

```

## 25.9.5 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 233](#).

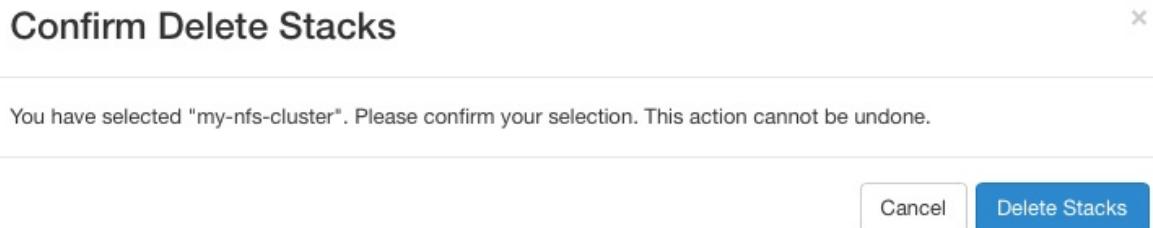


Figure 233: Delete Stacks

The template for the [Hello World complex appliance](#) is reproduced below. 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 below: [Figure 234](#), as you can see, the two client instances are shown separately since each one is defined as a separate resource in the template.

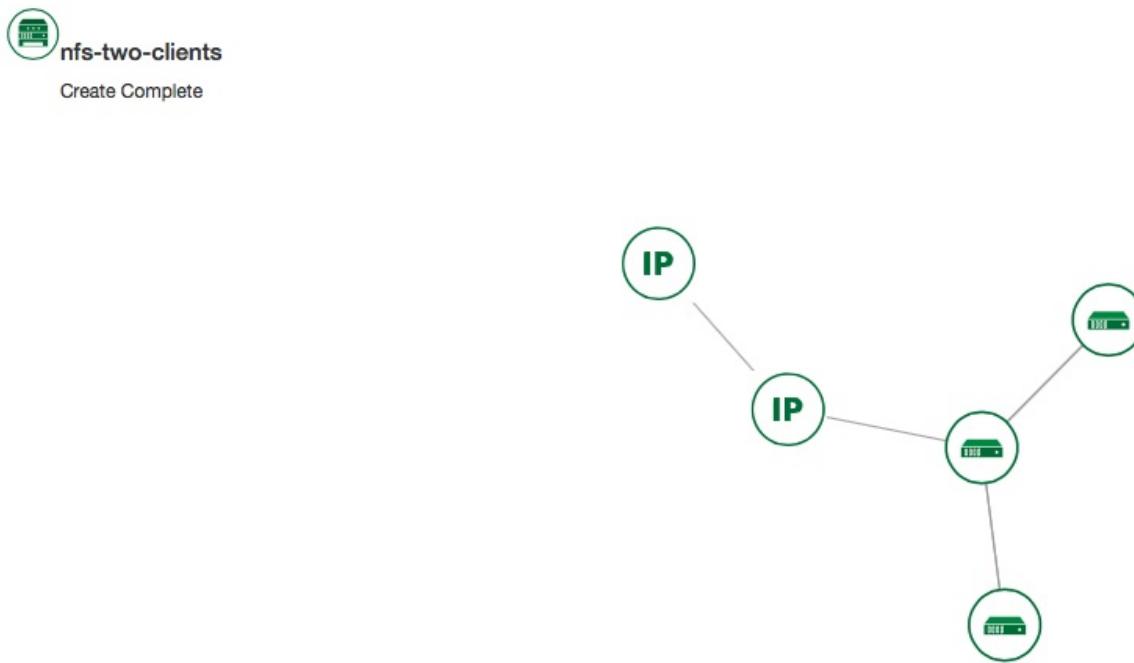


Figure 234: NFS with Two Clients

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 below: [Figure 235](#), shows the resulting outputs as viewed from the web interface. Of course IP addresses will be specific to each deployment.

## Outputs

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>client_ips</code> | Private IP address of the NFS clients       |
|                         | [<br>"10.140.82.20",<br>"10.140.82.19"<br>] |
| <code>server_ip</code>  | Public IP address of the NFS server         |
|                         | 130.202.88.157                              |

Figure 235: 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.

## 25.9.6 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

### 25.9.6.1 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
```

### 25.9.6.2 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.
```

### 25.9.6.3 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.

#### 25.9.6.4 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.

#### 25.9.6.5 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.

#### 25.9.7 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 236](#)).

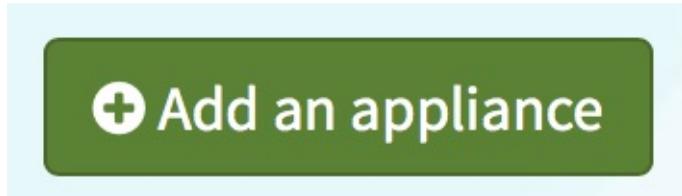


Figure 236: Add an Appliance

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!

## 25.9.8 Advanced topics

### 25.9.8.1 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 pattern 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 below illustrates how it works. It launches a group of instances that will automatically populates 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 $(facter ipaddress); echo -n ' ' ; echo $(facter hostname)) > ${heat_output}

 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.

## 25.10 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 237](#).

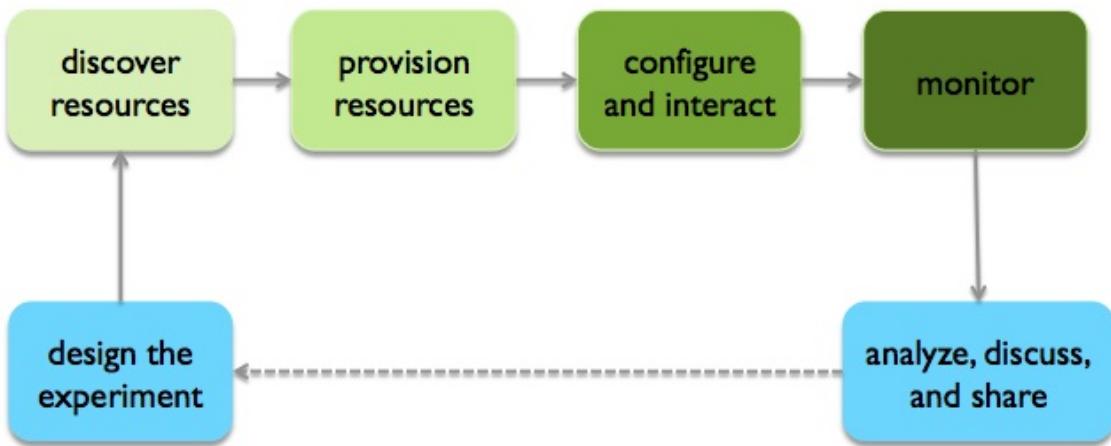


Figure 237: Experimenting with Bare Metal resources on Chameleon cloud

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 guides above:

- the [Chameleon Object Store](#),
- [network isolation for bare metal](#).

## **25.11 CHAMELEON CLOUD FREQUENTLY ASKED QUESTIONS**

---



### **25.11.1 Appliances**

#### **25.11.1.1 What is an appliance?**

An appliance is an application packaged together with the environment that this application requires. For example, an appliance can consist 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](#).

### **25.11.1.2 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 as appliances contributed by users.

### **25.11.1.3 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](#).

#### **25.11.1.4 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.

#### **25.11.1.5 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.

#### **25.11.1.6 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.

### **25.11.2 Bare Metal Troubleshooting**

#### **25.11.2.1 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](#).

## 25.11.3 OpenStack KVM Troubleshooting

### 25.11.3.1 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](#).

### 25.11.3.2 Why cannot I ping or SSH to my instance?

While the possibility that the system is being taking over by nanites should not be discounted too easily, it is always prudent to first check for the following issues:

- 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 above (and any evidence of nanites you find as well).



### 26.1 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

## **26.2 NETWORK**

---

Bridged Networking

TBD

External Network

TBD

Internal Network

TBD

Local Bridge

TBD

Network Address Translation (NAT)

TBD

## **26.3 STORAGE**

---

Block Device

TBD

Virtual Disk

TBD

Raw Disk

TBD



This section contains section, chapters and examples from students that have not yet been integrated in the overall structure of the document. Please note that we may move them at any time. Make sure to stay up to date with the master.

We recommend sections to be at least 300 words

Chapters should be much longer and comprehensive

Example need to be short, but complete. IN acse you need code to be submitted you can do this in a code directory

### 27.1 Box O ?



what is box

REST

- <https://developer.box.com/reference>

Limitations:

- Box has some very nasty limitations, please research them if you can
- examples: limitation in total download
- example: limitation in organization when a team member leaves. e.g. repos have to be set up for the organization and not as individual file owner. What will happen if the files do not belong to the organization.
- also think about what if my directory has 10, 10000, 100000 or 1M files.

Link:

- <http://opensource.box.com/box-python-sdk/tutorials/intro.html>
- <https://github.com/box/box-python-sdk/tree/1.5/demo>

Install:

```
pip install boxsdk
```

app.cfg:

```
* A Box client ID for a Box application
* The corresponding Box client secret
* A valid developer token for that application
```

code:

```
Import two classes from the boxsdk module - Client and OAuth2
from boxsdk import Client, OAuth2

Define client ID, client secret, and developer token.
CLIENT_ID = None
CLIENT_SECRET = None
ACCESS_TOKEN = None

Read app info from text file
with open('app.cfg', 'r') as app_cfg:
 CLIENT_ID = app_cfg.readline()
 CLIENT_SECRET = app_cfg.readline()
 ACCESS_TOKEN = app_cfg.readline()

Create OAuth2 object. It's already authenticated, thanks to the developer token.
oauth2 = OAuth2(CLIENT_ID, CLIENT_SECRET, access_token=ACCESS_TOKEN)

Create the authenticated client
client = Client(oauth2, LoggingNetwork())

Get information about the logged in user (that's whoever owns the developer token)
my = client.user(user_id='me').get()
print (my.name)
print (my.login)
print (my.avatar_url)

root_folder = client.folder('0')
root_folder_with_info = root_folder.get()

print (root_folder_with_info)
```

## Questions

- how to list all files in dir
- how to recursively iterate
- how to download each file when we iterate

### 27.1.1 boxpython

- <https://github.com/wesleyfr/boxpython>

```
from boxpython import BoxAuthenticateFlow, BoxSession, BoxError

flow = BoxAuthenticateFlow('\'<client_id\>' , '\'<client_secret\>')
flow.get_authorization_url()
'<https://www.box.com/api/oauth2/authorize?response_type=code&client_id=<client_id>&client_secret=<client_secret>>')

access_token, refresh_token =
flow.get_access_tokens('\'<auth_code\>')

def tokens_changed(refresh_token, access_token): ...
 save_to_file(refresh_token, access_token) ... \>\>\> box =
BoxSession('\'<client_id\>' , '\'<client_secret\>' , refresh_token,
access_token, tokens_changed)

box.get_folder_info(0)

box.download_file(11006194629, '/tmp/test_dl.txt')
```

### 27.1.2 Pybox

<https://github.com/hzheng/pybox>

## 27.2 COMPLIANCE AND GRID COMPUTING



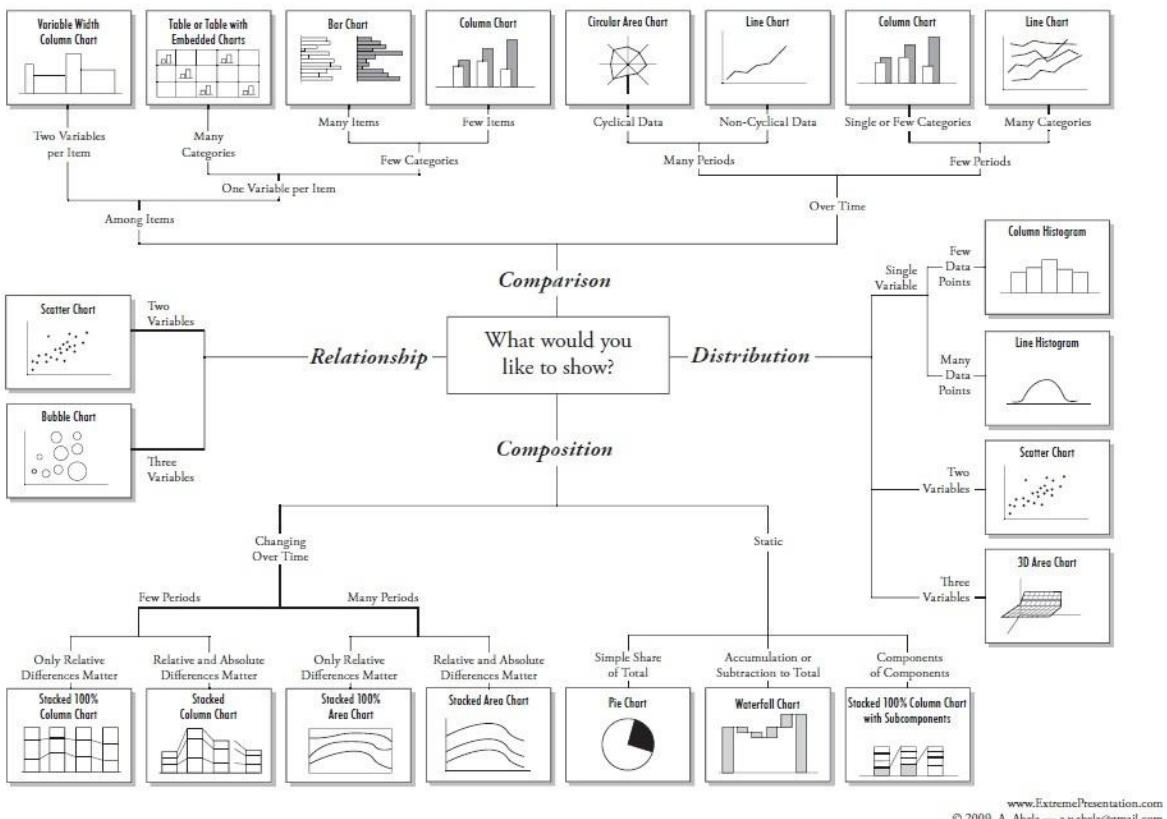
## 27.3 VISUALIZATION



All tools must be free.

### 27.3.1 Chart Types

## Chart Suggestions—A Thought-Starter



www.ExtremePresentation.com  
© 2009 A. Abela — a.abela@gmail.com

Figure 238: Visualisation

- <https://extremepresentation.typepad.com/files/choosing-a-good-chart-09.pdf>

### 27.3.2 D3 Gallery

- <https://github.com/d3/d3/wiki/Gallery>
- <https://c3js.org/examples.html>
- <http://nvd3.org/examples/index.html>

### 27.3.3 Matplotlib Gallery

TBD

### 27.3.4 Bokeh Gallery

- <https://bokeh.pydata.org/en/latest/docs/gallery.html>

### **27.3.5 R Gallery**

- <https://www.r-graph-gallery.com/>

### **27.3.6 Gnuplot**

- <http://www.gnuplot.info/screenshots/index.html#demos>

### **27.3.7 React**

- <https://uber.github.io/react-vis/examples/showcases/plots>
- <https://formidable.com/open-source/victory/gallery/>

### **27.3.8 Tablaeu**

Tablaue 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.

## **27.4 VISUAL STUDIO FOR CLOUD COMPUTING**



## **27.5 WINDOWS SUBSYSTEM FOR LINUX**





## REFERNCE



- [1] G. von Laszewski, F. Wang, H. Lee, H. Chen, and G. C. Fox, "Accessing Multiple Clouds with Cloudmesh," in Proceedings of the 2014 acm international workshop on software-defined ecosystems, 2014, pp. 21–28 [Online]. Available: <http://doi.acm.org/10.1145/2609441.2609638>
- [2] domo.com, "Data never sleeps 6.0." Image, Jun-2018 [Online]. Available: <https://www.domo.com/blog/wp-content/uploads/2018/06/18-domo-data-never-sleeps-6.png>
- [3] L. Lewis, "This is what happens in an internet minute." Web page, Apr-2018 [Online]. Available: <https://www.allaccess.com/merge/archive/28030/2018-update-what-happens-in-an-internet-minute>
- [4] visibletechnologies.com, "Big data 36 month." Image, Mar-2012 [Online]. Available: [https://blogs-images.forbes.com/christopherfrank/files/2012/03/VI\\_BigData\\_Graphi](https://blogs-images.forbes.com/christopherfrank/files/2012/03/VI_BigData_Graphi)
- [5] K. Heslin, "Proper data center staffing is key to reliable operations." Web page, Mar-2015 [Online]. Available: <https://journal.uptimeinstitute.com/data-center-staffing/>
- [6] D. Bouley, "Estimating a data center's electrical carbon footprint," Schneider Electric – Data Center Science Center, Report 66, 2011 [Online]. Available: [http://www.apc.com/salestools/DBOY-7EVHLH/DBOY-7EVHLH\\_R0\\_EN.pdf](http://www.apc.com/salestools/DBOY-7EVHLH/DBOY-7EVHLH_R0_EN.pdf)
- [7] Indiana State, "Indiana - state energy profile overview - u.s. Energy information." Web page, Apr-2018 [Online]. Available: <https://www.eia.gov/state/?sid=IN>
- [8] T. R. Furlani et al., "Using xdmod to facilitate xsede operations, planning and analysis," in Proceedings of the conference on extreme science and engineering discovery environment: Gateway to

discovery, 2013, pp. 46:1–46:8 [Online]. Available: <http://doi.acm.org/10.1145/2484762.2484763>

[9] F. Wang, G. von Laszewski, G. C. Fox, T. R. Furlani, R. L. DeLeon, and S. M. Gallo, "Towards a scientific impact measuring framework for large computing facilities - a case study on xsede," in Proceedings of the 2014 annual conference on extreme science and engineering discovery environment, 2014, pp. 25:1–25:8 [Online]. Available: <http://cgl.soic.indiana.edu/publications/Metrics2014.pdf>

[10] G. von Laszewski, "FutureGrid cloud metrics." Web page, Sep-2014 [Online]. Available: <http://archive.futuregrid.org/metrics/html/results/2014-Q3/reports/rst/india-All.html>

[11] Amazon, "Amazon data centers." Web Page, Jan-2019 [Online]. Available: <https://aws.amazon.com/compliance/data-center/data-centers/>

[12] Amazon, "AWS global infrastructure." Web Page, Jan-2019 [Online]. Available: <https://aws.amazon.com/about-aws/global-infrastructure/>

[13] Microsoft, "Azure regions." Web page, Jan-2019 [Online]. Available: <https://azure.microsoft.com/en-us/global-infrastructure/regions/>

[14] Google, "Google locations." Web page, Jan-2019 [Online]. Available: <https://www.google.com/about/datacenters/inside/locations/index.htm>

[15] Google, "Efficiency: How we do it." Web Page, Jan-2019 [Online]. Available: <https://www.google.com/about/datacenters/efficiency/internal/>

[16] A. Shehabi et al., "United states data center energy usage report," Lawrence Berkeley National Laboratory, Report DE-AC02-05CH1131, LBNL-1005775, Jun. 2016 [Online]. Available:

<https://escholarship.org/uc/item/84p772fc>

- [17] Microsoft, "Microsoft leona philpot." Web page [Online]. Available: <https://news.microsoft.com/features/microsoft-research-project-puts-cloud-in-ocean-for-the-first-time/>
- [18] Microsoft, "Microsoft northern isles." Web page [Online]. Available: <https://news.microsoft.com/features/under-the-sea-microsoft-tests-a-datacenter-thats-quick-to-deploy-could-provide-internet-connectivity-for-years/>
- [19] New York times, "Microsoft underwater datacenter." Web page [Online]. Available: <https://www.nytimes.com/2016/02/01/technology/microsoft-plumbs-oceans-depths-to-test-underwater-data-center.html>
- [20] I. Foster and C. Kesselman, Eds., The grid: Blueprint for a new computing infrastructure. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.
- [21] National Institute of Standars, "NIST big data public working group." [Online]. Available: <https://bigdatawg.nist.gov/>
- [22] S. J. Bigelow, "Full virtualization vs. Paravirtualization: What are the key differences?" Wegrb page, Sep-2018 [Online]. Available: <https://searchservervirtualization.techtarget.com/answer/Full-virtualization-vs-paravirtualization-What-are-the-key-differences>
- [23] Wikipedia, "Input-output memory management unit." Web page [Online]. Available: [https://en.wikipedia.org/wiki/Input%E2%80%93output\\_memory\\_ma](https://en.wikipedia.org/wiki/Input%E2%80%93output_memory_ma)
- [24] Wikipedia, "Input-output memory management unit." Web page [Online]. Available: [https://en.wikipedia.org/wiki/X86\\_virtualization#I.2FO\\_MMU\\_virtualiza](https://en.wikipedia.org/wiki/X86_virtualization#I.2FO_MMU_virtualiza)  
[Vi\\_and\\_Intel\\_VT-d.29](https://en.wikipedia.org/wiki/X86_virtualization#Vi_and_Intel_VT-d.29)
- [25] "Libvirt." Web page [Online]. Available:

[https://www.suse.com/documentation/sles11/book\\_kvm/data/cha\\_lib](https://www.suse.com/documentation/sles11/book_kvm/data/cha_lib)

[26] J. Guerrag, "Difference between kvm and qemu." Forum post, Dec-2010 [Online]. Available:

<https://serverfault.com/questions/208693/difference-between-kvm-and-qemu>

[27] Wikipedia, "VMWare." Web page, Sep-2018 [Online]. Available: <https://en.wikipedia.org/wiki/VMware>

[28] "Wine – wine is not an emulator." Web Page, Sep-2018 [Online]. Available: <https://www.winehq.org/>

[29] Stackoverflow, "What are the differences between qemu and virtualbox?" Forum post [Online]. Available: <https://stackoverflow.com/questions/43704856/what-are-the-differences-between-qemu-and-virtualbox>

[30] "Hadoop mapreduce." [Online]. Available: [https://www.edureka.co/blog/mapreduce-tutorial/?utm\\_source=youtube&utm\\_campaign=mapreduce-tutorial-161216-wr&utm\\_medium=description](https://www.edureka.co/blog/mapreduce-tutorial/?utm_source=youtube&utm_campaign=mapreduce-tutorial-161216-wr&utm_medium=description)

[31] "Hadoop mapreduce." [Online]. Available: <https://www.youtube.com/watch?v=SqvAaB3vK8U&list=WL&index=25&t=2547s>

[32] "Apache mapreduce." [Online]. Available: <https://www.ibm.com/analytics/hadoop/mapreduce>

[33] "MapReduce." [Online]. Available: <https://en.wikipedia.org/wiki/MapReduce>

[34] "Hadoop mapreduce." [Online]. Available: [https://www.tutorialspoint.com/hadoop/hadoop\\_mapreduce.htm](https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm)

[35] A. Khan, "Hadoop and spark." [Online]. Available: <https://www.quora.com/What-is-the-difference-between-Hadoop-and-Spark>. [Accessed: 03-Sep-2017]

[36] “Apache spark vs hadoop mapreduce.” [Online]. Available: <https://data-flair.training/blogs/apache-spark-vs-hadoop-mapreduce/>

[37] AWS, “Setup prerequisites for your cluster.” Web Page, Mar-2018 [Online]. Available: <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-prerequisites.html>

[38] AWS, “How do i create an s3 bucket?” Web Page, Mar-2018 [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-bucket.html>

[39] AWS, “Create a key pair using amazon ec2.” Web Page, Mar-2018 [Online]. Available: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair>

[40] AWS, “Launch sample emr cluster.” Web Page, Mar-2018 [Online]. Available: <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-launch-sample-cluster.html>

[41] AWS, “Process your sample data.” Web Page, Mar-2018 [Online]. Available: <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-process-sample-data.html>

[42] AWS, “Reset your environment.” Web Page, Mar-2018 [Online]. Available: <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-reset-environment.html>

[43] Docker, “Overview of docker hub.” Web Page, Mar-2018 [Online]. Available: <https://docs.docker.com/docker-hub/>

[44] S. Bhartiya, “How to use dockerhub.” Blog, Jan-2018 [Online]. Available: <https://www.linux.com/blog/learn/intro-to->

## [linux/2018/1/how-use-dockerhub](https://linux/2018/1/how-use-dockerhub)

- [45] Docker, "Repositories on docker hub." Web Page, Mar-2018 [Online]. Available: <https://docs.docker.com/docker-hub/repos/>
- [46] R. Irani, "Docker tutorial series-part 4-docker hub." Blog, Jul-2015 [Online]. Available: <https://rominirani.com/docker-tutorial-series-part-4-docker-hub-b51fb545dd8e>
- [47] A. Ellis, "Introducing functions as a service (openfaas)." Web page, Aug-2017 [Online]. Available: <https://blog.alexellis.io/introducing-functions-as-a-service/>
- [48] C. WODEHOUSE, "Should you use mongodb? A look at the leading nosql database." Web Page, 2018 [Online]. Available: <https://www.upwork.com/hiring/data/should-you-use-mongodb-a-look-at-the-leading-nosql-database/>
- [49] Guru99, "Introduction to mongodb." Web Page, 2018 [Online]. Available: <https://www.guru99.com/mongodb-tutorials.html#1>
- [50] MongoDB, "Https://www.mongodb.com/." Web Page, 2018 [Online]. Available: <https://docs.mongodb.com/manual/introduction/>
- [51] M. Papiernik, "How to install mongodb on ubuntu 18.04." Web Page, Jun-2018 [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-18-04>
- [52] J. Ellingwood, "Initial server setup with ubuntu 18.04." Web Page, Apr-2018 [Online]. Available: <https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-18-04>
- [53] MongoDB, Databases and collections, 4.0 ed. New York, New York, USA: MongoDB Inc, 2008 [Online]. Available: <https://docs.mongodb.com/manual/core/databases-and-collections/>
- [54] J. M. Craig Buckler, "Using joins in mongodb nosql databases."

Web Page, Sep-2016 [Online]. Available:  
<https://www.sitepoint.com/using-joins-in-mongodb-nosql-databases/>

[55] MongoDB, Lookup (aggregation), 3.2 ed. New York City, New York, United States: MongoDB Inc, 2008 [Online]. Available:  
<https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/>

[56] MongoDB, MongoDB package components - mongoexport, 4.0 ed. New York City, New York, United States: MongoDB Inc, 2008 [Online]. Available:  
<https://docs.mongodb.com/manual/reference/program/mongoexport/>

[57] MongoDB, Security, 4.0 ed. New York City, New York, United States: MongoDB Inc, 2008 [Online]. Available:  
<https://docs.mongodb.com/manual/security/>

[58] MongoDB, “MongoDB atlas.” Web Page, 2018 [Online]. Available:  
<https://www.mongodb.com/cloud/atlas>

[59] I. MongoDB, “PyMongo 3.7.1 documentation.” Web Page, 2008 [Online]. Available: <https://api.mongodb.com/python/current/api>

[60] A. J. J. Davis, “Announcing pymongo3.” Web Page, Apr-2015 [Online]. Available: <https://emptysqua.re/blog/announcing-pymongo-3/>

[61] M. Dirolf, “PyMongo.” Web Page, Jul-2018 [Online]. Available:  
<https://github.com/mongodb/mongo-python-driver>

[62] N. Leite, “MongoDB and python.” Web Page, Mar-2015 [Online]. Available: <https://www.slideshare.net/NorbertoLeite/mongodb-and-python>

[63] V. Oleynik, “How do you use mongodb with python?” Web Page, Mar-2017 [Online]. Available: <https://gearheart.io/blog/how-do-you-use-mongodb-with-python/>

[64] I. MongoDB, “Installing / upgrading.” Web pages, 2008 [Online]. Available: <http://api.mongodb.com/python/current/installation.html>

- [65] R. Python, "Introduction to mongodb and python." Web Page, 2016 [Online]. Available: <https://realpython.com/introduction-to-mongodb-and-python/>
- [66] W3Schools, "Python mongodb create database." Web Page, 1999 [Online]. Available: [https://www.w3schools.com/python/python\\_mongodb\\_create\\_db.asp](https://www.w3schools.com/python/python_mongodb_create_db.asp)
- [67] I. MongoDB, "PyMongo 3.7.1 documentation." Web Page, 2008 [Online]. Available: <https://api.mongodb.com/python/current/tutorial.html>
- [68] N. O'Higgins, PyMongo & python. O'Reilly, 2011 [Online]. Available: <http://img105.job1001.com/upload/adminnew/2015-04-07/1428393873-MHKX3LN.pdf>
- [69] I. MongoDB, "PyMongo 3.7.1 documentation." Web Page, 2008 [Online]. Available: <https://api.mongodb.com/python/current/examples/aggregation.html>
- [70] MongoDB, "PyMongo 3.7.2 documenation." Web Page, 2008 [Online]. Available: <https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/>
- [71] MongoDB, "PyMongo 3.7.2 documenation." Web Page, 2008 [Online]. Available: <https://docs.mongodb.com/manual/core/map-reduce/>
- [72] MongoDB, "PyMongo v2.0 documentation." Web Page, 2008 [Online]. Available: [https://api.mongodb.com/python/2.0/examples/map\\_reduce.html](https://api.mongodb.com/python/2.0/examples/map_reduce.html)
- [73] MongoDB, "PyMongo 3.7.2 documenation." Web Page, 2008 [Online]. Available: <https://api.mongodb.com/python/current/examples/copydb.html>
- [74] MongoEngine, "MongoEngine user documentation." Web Page,

2009 [Online]. Available: <http://docs.mongoengine.org/>

[75] Wikipedia, “Object-relational mapping.” Web Page, May-2009 [Online]. Available: [https://en.wikipedia.org/wiki/Object-relational\\_mapping](https://en.wikipedia.org/wiki/Object-relational_mapping)

[76] MongoDB, “Flask-mongoengine.” Web Page, 2008 [Online]. Available: <http://docs.mongoengine.org/guide/defining-documents.html>

[77] MongoEngine, “User guide: Document instances.” Web Page, 2009 [Online]. Available: <http://docs.mongoengine.org/guide/document-instances.html>

[78] MongoEngine, “2.1 installing mongoengine.” Web Page, 2009 [Online]. Available: <http://docs.mongoengine.org/guide/installing.html>

[79] MongoEngine, “2.2 connection to mongodb.” Web Page, 2009 [Online]. Available: <http://docs.mongoengine.org/guide/connecting.html>

[80] MongoEngine, “User guide 2.5. Querying the database.” Web Page, 2009 [Online]. Available: <http://docs.mongoengine.org/guide/querying.html>

[81] wikipedia, “Flask (web framework).” Web Page, 2010 [Online]. Available: [https://en.wikipedia.org/wiki/Flask\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))

[82] MongoDB, “Flask-pymongo.” Web Page, 2008 [Online]. Available: <https://flask-pymongo.readthedocs.io/en/latest/>

[83] MongoDB, “Flask mongoalchemy.” Web Page, 2008 [Online]. Available: <https://pythonhosted.org/Flask-MongoAlchemy/>

[84] MongoDB, “Flask-mongoengine.” Web Page, 2008 [Online]. Available: <http://docs.mongoengine.org/projects/flask-mongoengine/en/latest/>

[85] Wikipedia, “Flask (web framework).” Web Page, Oct-2018 [Online].

Available: [https://en.wikipedia.org/wiki/Flask\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))

[86] Golang, “The go programming language.” Web page, Sep-2018 [Online]. Available: <https://golang.org/doc/>

[87] C. A. R. Hoare, Communicating sequential processes. Electronit version of Prentice Hall International, 1985 [Online]. Available: <http://www.usingcsp.com/cspbook.pdf>

[88] D. Pountain and D. May, A tutorial introduction to occam programming. New York, NY, USA: McGraw-Hill, Inc., 1987 [Online]. Available: <http://www.transputer.net/oBooks/72-occ-046-00/tuinocc.pdf>

[89] D. C. Hyde, Introduction to the programming language occam. Bucknell University, 1995 [Online]. Available: <http://www.cs.otago.ac.nz/cosc441/occam.pdf>

[90] Tiobe, “TIOBE index.” Web page, Sep-2018 [Online]. Available: <https://www.tiobe.com/tiobe-index/>

[91] K. Seguin, “The little go book.” Web page, Jan-2018 [Online]. Available: <https://github.com/karlseguin/the-little-go-book>

[92] Stefan Nilsson, “Algorithms to go.” Web page [Online]. Available: <https://yourbasic.org/>

[93] P. Caponetti, “Why mqtt is the protocol of choice for the iot.” xively.com blog website, Aug-2017 [Online]. Available: <http://blog.xively.com/why-mqtt-is-the-protocol-of-choice-for-the-iot/>

[94] hivemq, “Intrwebsite mqtt.” hivemq website [Online]. Available: <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt>

[95] Wikipedia, “MQTT — wikipedia, the free encyclopedia.” Nov-2017 [Online]. Available: <https://en.wikipedia.org/w/index.php?title=MQTT&oldid=808683219>

[96] Mqtt, "Mqtt official website." mqtt official website [Online]. Available: <http://mqtt.org/>

[97] eclipse, "Mqtt broker." eclipse mosquitto website [Online]. Available: <https://mosquitto.org/>

[98] random nerds tutorial, "What is mqtt and how it works." random nerds website [Online]. Available: <https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/>

[99] H. mq, "MQTT essentials part 2: Publish and subscribe." HiveMQ website [Online]. Available: <https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe>

[100] eclipse paho, "Python client - documentation." eclipse paho website [Online]. Available: <https://www.eclipse.org/paho/clients/python/docs/>

[101] H. MQ, "MQTT essentials part 6: Quality of service 0, 1 and 2." HiveMQ website [Online]. Available: <https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels>

[102] T. Ouska, "Transport-level security tradeoffs using mqtt." iot design website, Feb-2016 [Online]. Available: <http://iotdesign.embedded-computing.com/guest-blogs/transport-level-security-tradeoffs-using-mqtt/>

[103] H. Mq, "MQTT security fundamentals: TLS / ssl." hive mq website [Online]. Available: <https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl>

[104] I. Craggs, "MQTT security: Who are you? Can you prove it? What can you do?" IBM developer works website, Mar-2013 [Online]. Available:

[https://www.ibm.com/developerworks/community/blogs/c565c720-fe84-4f63-873f-607d87787327/entry/mqtt\\_security?lang=en](https://www.ibm.com/developerworks/community/blogs/c565c720-fe84-4f63-873f-607d87787327/entry/mqtt_security?lang=en)

[105] hive mq, "MQTT security fundamentals: OAuth 2.0 and mqtt." hivemq website [Online]. Available: <https://www.hivemq.com/blog/mqtt-security-fundamentals-oauth-2-0-mqtt>

[106] apache, "Apache storm." apache storm website [Online]. Available: <http://storm.apache.org/>

[107] A. storm, "Storm mqtt integration." Apache storm website [Online]. Available: <http://storm.apache.org/releases/1.1.0/storm-mqtt.html>

[108] Wikipedia, "Storm (event processor) — wikipedia, the free encyclopedia." 2017 [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Storm\\_\(event\\_processor\)&oldid=808771136](https://en.wikipedia.org/w/index.php?title=Storm_(event_processor)&oldid=808771136)

[109] elastic.io, "ELK stack." elastic.io website [Online]. Available: <https://www.elastic.co/products>

[110] smart factory, "Storing iot data using open source. MQTT and elasticsearch - tutorial." smart factory website, Oct-2016 [Online]. Available: <https://smart-factory.net/mqtt-elasticsearch-setup/>

[111] smart factory, "MQTT and kibana - open source graphs and analysis for iot." smart factory website, May-2016 [Online]. Available: <https://smart-factory.net/mqtt-and-kibana-open-source-graphs-and-analysis-for-iot/>

[112] erlang-mqtt, "Erlang mqtt broker." wmqtt website [Online]. Available: <http://emqtt.io/docs/v2/index.html>

[113] D. Industries, "Grovepi." Dexter Industries website, 2017 [Online]. Available: <https://www.dexterindustries.com/grovepi/>

[114] S. Studio, "Grove relay." seed studio website, Oct-2017 [Online]. Available: <http://wiki.seeed.cc/Grove-Relay/>

[115] seed studio, "Grove led socket kit." Seed studio website, Oct-

2017 [Online]. Available: [http://wiki.seeed.cc/Grove-LED\\_Socket\\_Kit/](http://wiki.seeed.cc/Grove-LED_Socket_Kit/)

[116] cloudmesh, “Cloudmesh.pi.” github, Oct-2017 [Online]. Available: <https://github.com/cloudmesh/cloudmesh.pi>

[117] Facebook, “Introduction to graphql.” Web page [Online]. Available: <https://graphql.org/learn/>

[118] Django Software Foundation, “Django web framework.” Web page [Online]. Available: <https://www.djangoproject.com/>

[119] João Angelo, “JWT tokens.” Web page [Online]. Available: <https://stackoverflow.com/a/39914013>

[120] Clay Allsopp, “GraphQL and authentication.” Web page [Online]. Available: <https://medium.com/the-graphqlhub/graphql-and-authentication-b73aed34bbeb>

[121] Github, “Github api v4.” Web page [Online]. Available: <https://developer.github.com/v4/>

[122] Jonatas Baldin, “Introduction to graphql python implementation.” Web page [Online]. Available: <https://www.howtographql.com/graphql-python/0-introduction/>

[123] Apache Avro, “Apache avro 1.8.2 documentation.” Web Page, Feb-2017 [Online]. Available: <http://avro.apache.org/docs/1.8.2/index.html>

[124] Apache Avro, “Apache avro 1.8.2 getting started (python).” Web Page, Feb-2017 [Online]. Available: <http://avro.apache.org/docs/1.8.2/gettingstartedpython.html>

[125] Apache Avro, “Apache avro 1.8.2 getting started (java).” Web Page, Feb-2017 [Online]. Available: <http://avro.apache.org/docs/1.8.2/gettingstartedjava.html>

[126] Apache Avro, “Apache avro 1.8.2 hadoop mapreduce guide.” Web Page, Feb-2017 [Online]. Available:

<http://avro.apache.org/docs/1.8.2/mr.html>

[127] Apache Avro, “Apache avro 1.8.2 specification.” Web Page, Feb-  
2017 [Online]. Available:

[http://avro.apache.org/docs/1.8.2/spec.html#schema\\_record](http://avro.apache.org/docs/1.8.2/spec.html#schema_record)