Intelligent Systems Engeneering
e222

Geoffrey C. Fox

Gregor von Laszewski

Editor

laszewski@gmail.com

https://cloudmesh-community.github.io/pub/vonlaszewski-222.epub

November 23, 2019 - 05:18 AM

INTELLIGENT SYSTEMS ENGENEERING

Geoffrey C. Fox Gregor von Laszewski

INTELLIGENT SYSTEMS ENGENEERING

1 PREFACE

Sat Nov 23 05:18:45 EST 2019 ☁

1.1 DISCLAIMER ☁

This book has been generated with [Cyberaide Bookmanager](#).

Bookmanager is a tool to create a publication from a number of sources on the internet. It is especially useful to create customized books, lecture notes, or handouts. Content is best integrated in markdown format as it is very fast to produce the output.

Bookmanager has been developed based on our experience over the last 3 years with a more sophisticated approach. Bookmanager takes the lessons from this approach and distributes a tool that can easily be used by others.

The following shields provide some information about it. Feel free to click on them.

`pypi v0.2.28` `License Apache 2.0` `python 3.7` `format wheel` `status stable` `build unknown`

1.1.1 Acknowledgment

If you use bookmanager to produce a document you must include the following acknowledgement.

> *"This document was produced with Cyberaide Bookmanager developed by Gregor von Laszewski available at https://pypi.python.org/pypi/cyberaide-bookmanager. It is in the responsibility of the user to make sure an author acknowledgement section is included in your document. Copyright verification of content included in a book is responsibility of the book editor."*

The bibtex entry is

```
@Misc{www-cyberaide-bookmanager,
  author =   {Gregor von Laszewski},
```

```
  title =     {{Cyberaide Book Manager}},
  howpublished = {pypi},
  month =     apr,
  year =      2019,
  url={https://pypi.org/project/cyberaide-bookmanager/}
}
```

1.1.2 Extensions

We are happy to discuss with you bugs, issues and ideas for enhancements. Please use the convenient github issues at

- https://github.com/cyberaide/bookmanager/issues

Please do not file with us issues that relate to an editors book. They will provide you with their own mechanism on how to correct their content.

1.2 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, Brijesh, Dave DeMeulenaere, De'Angelo Rutledge, Eliyah Ben Zayin, Eric Bower, Fugang Wang, Geoffrey C. Fox, Gerald Manipon, Gregor von Laszewski, Hyungro Lee, Ian Sims, IzoldaIU, Javier Diaz, Jeevan Reddy Rachepalli, Jonathan Branam, Juliette Zerick, Keith Hickman, Keli Fine, Kenneth Jones, Mallik Challa, 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, Sheri Sanders, Shivani Katukota, Silvia Karim, Swarnima H. Sowani,*

Tharak Vangalapat, Tim Whitson, Tyler Balson, Vafa Andalibi, Vibhatha Abeykoon, Vineet Barshikar, Yu Luo, ahilgenkamp, aralshi, azebrowski, bfeng, brandonfischer99, btpope, garbeandy, harshadpitkar, himanshu3jul, hrbahramian, isims1, janumudvari, joshish-iu, juaco77, karankotz, keithhickman08, kkp, mallik3006, manjunathsivan, niranda perera, qianqian tang, rajni-cs, rirasto, sahancha, shilpasingh21, swsachith, toshreyanjain, trawat87, tvangalapat, varunjoshi01, vineetb-gh, xianghang mi, zhengyili4321

2 SYLLABUS

2.1 E222: INTELLIGENT SYSTEMS ENGENEERING II ☁

In this undergraduate course students will be familiarized with different specific applications and implementations of intelligent systems and their use in desktop and cloud solutions.

- Piazza: Link
- Registrar: Link
- Lecture Notes: ePub
- Indiana University
- Faculty: Geoffrey C. Fox
- Credits: 3
- Hardware: You will need a computer to take this class, a phone, tablet, or chrome book is not sufficient.
- 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 IU computers or cloud virtual machines. The later have to be requested.
- Course Description: Link

This is an introductory class. In case you like to do research and more advanced topics, consider taking an independent study with Dr. Fox or Dr. von Laszewski.

An introduction video is available at:

222 Class Introduction and Management

2.1.1 Teaching and learning methods

- Lectures
- Assignments including specific lab activities

- Final project

2.1.2 Representative bibliography

1. Cloud Computing for Science and Engineering By Ian Foster and Dennis B. Gannon

- https://mitpress.mit.edu/books/cloud-computing-science-and-engineering

2. (This document) Handbook of Clouds and Big Data, Gregor von Laszewski, Geoffrey C. Fox, and Judy Qiu, Fall 2017, https://tinyurl.com/vonLaszewski-handbook
3. Use Cases in Big Data Software and Analytics Vol. 1, Gregor von Laszewski, Fall 2017, https://tinyurl.com/cloudmesh/vonLaszewski-i523-v1.pdf
4. Use Cases in Big Data Software and Analytics Vol. 2, Gregor von Laszewski, Fall 2017, https://tinyurl.com/cloudmesh/vonLaszewski-i523-v2.pdf
5. Use Cases in Big Data Software and Analytics Vol. 3, Gregor von Laszewski, Fall 2017, https://tinyurl.com/vonLaszewski-projects-v3
6. Big Data Software Vol 1., Gregor von Laszewski, Spring 2017, https://github.com/cloudmesh/sp17-i524/blob/master/paper1/proceedings.pdf
7. Big Data Software Vol 2., Gregor von Laszewski, Spring 2017,

- https://github.com/cloudmesh/sp17-i524/blob/master/paper2/proceedings.pdf

8. Big Data Projects, Gregor von Laszewski, Spring 2017,

- https://github.com/cloudmesh/sp17-i524/blob/master/project/projects.pdf

9. Gregor von Laszewski, Geoffrey C. Fox, Cloud Computing and Big Data http://cyberaide.org/papers/vonLaszewski-bigdata.pdf
10. Introduction to Python for cloud Computing https://laszewski.github.io/book/python/

2.1.3 Grading

Grade Item	Percentage
Assignments	30%
Final Project	60%
Participation	10%

2.1.4 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.

2.1.5 Other classes I423, I523, I524, B649, E516, E616

IU offers other undergraduate classes in this topic area such as I423. If you are interested in taking it, please see when they are taught. Additional graduate level classes related which can also be taken only by special permission including:

- CSCI B-649 Cloud Computing is the same as E516 but for computer science students.
- I524 is the same as E516 but for Data Engineering Students
- E516 Introdcustion to CLoud COmputing and CLoud Engeneering

All of these classes are project based and require a significant and consistent effort of time on your side.

2.1.6 Communication

To ask for help use piazza:

- [Piazza Resources](#)
- [Piazza Questions](#)

Please do not use CANVAS for communicating with us. Use Piazza. Make sure you have access to Piazza, while posting your formal Bio.

2.1.6.1 How to take this class

This class is an undergraduate class that contains two sections that you must attend.

In this document we will introduce you theoretically to some concepts that are important for this class. This is done either through lectures, written material, or pointers to Web resources. You are responsible to

1. listen to the online lectures and understand them.
2. identify additional material that may help you in understanding the lectures. This could include additional resources on the internet
3. Contribute to the material by correcting errors and updates you may find.

Please note that we try to keep the material up to date with your help. However, in our field software and documentation changes quickly and if you identify updated material we expect that you help us fixing it. You will get credit doing so.

To allow you to be most flexible in taking this class, we certainly allow you to work ahead. Thus you can use all but the in person lectures ahead of time. The Syllabus will clearly identify which material is available. Note that the book may include sections that are not marked in the syllabus. You do not have to read such sections.

Please note that this class does not have *small* assignments and any assignment is likely to take you a significant amount of time. Thus it is advisable that you start your assignments early and make sure you do not do them in the last week before the assignment is due. This contrasts other undergraduate classes, that may focus on the assignment of a number of toy exerises. Instead we will work throughout the entire semester towards a project you will conduct. In order to make it earlier for you, we will introduce graded checkpoints of all large assignments. The grades for these checkpoints are final and can not be improved by work done later. Also here please be advised that some may take several weeks to conduct and it is your responsibility to devote enough time to these activities.

To asure progress, you will have to manage a notebook.md file in your github

directory (that we will create for you) in which you will update your weekly progress. If you miss a lecture, it is in your responsibility to inform yourself what was being taught. Attendance and participation will be graded as well as the update to notebook.md will be graded.

In the following calendar we put in the last day of the week when the assignments are typically due

2.1.7 Covered Topics

As part of this class you will have to explore the following topics. These topics are either included in this document, or we are pointing you within this document to other documents with the information.

If we forgot anything let us know. The order of the lectures and the lecture material are subject to change as we see fit.

⚠ *This weekly Agenda will be updated every week. Yo are required to check in every week for updates. At this time we have included an approximate weekly agenda.*

To see the differences to previous versions of this document, you can look at:

- https://github.com/cloudmesh-community/book/commits/master/chapters/e222-syllabus.md

To see if checkins succeed you can look at:

- https://circleci.com/gh/cloudmesh-community/book

Currently, the topics covered in the class include the following.

2.1.7.1 Week 1. Overview of this Class

We will provide an overview of this class.

Logistic: Get familiar with the class structure.

Read: Preface; Class Overview; Start reviewing your python knowledge

Assignment Accounts: Find a computer you can do the class programming on (tablet and chrombook will not suffice). Get an account on piazza.com with your **???** name Get an account on github.com (This is NOT the IU github) and apply there for a github username. Post the username into a form that will be send to you. Make sure that the account you send us is your github.com account. This is a graded assignment that must be completed in the first week of class

This must be completed in the first week by Friday. (Survey will be posted on Piazza).

Assignment notebook: Once you get your github directory, update the file `notebook.md`. Mind the spelling notebook is lowercase. Use simple markdown bullet lists to record your activities.

Assignment Development Environment: (Multiweek assignment, to be completed in the firts month) It is important that you have a development environment to conduct the class assignments. We recommend that you use virtual box and use ubuntu. We have provided an extensive set of material for you to achieve this in this document. Please consult additional resources form the Web and utilize the Lab hours.

2.1.7.2 Week 1 and 2. Review of Python for Intelligent Systems Engineering

- Theory: basic Python Language
- Theory: pyenv, setup.py, modules

- Practice: Living without anaconda

- Python specific topics include:

  - Assignment: Install Python and use it throughut the semester
  - Why not anaconda?
  - Using python 3.7
  - pip

- Language
- Numpy
- Scipy
- OpenCV
- ScikitLearn

Report: Create an empty report based on our template in github. The TAs may do this with you in the Lab.

Github Pull Requests: Find a spelling error in the class material and create a pull request to correct it.

2.1.7.3 Week 2. Review of Linux shell for OSX, Linux, and Windows

- Theory: Basic Linux Shell
- Practice: SSH
- Assignment: ssh key generation on your computer, upload to github.com

2.1.7.4 Week 3. Introduction to REST

- Theory: Overview of REST, Eve, OpenAPI
- Practice: develop a REST service with OpenAPI

Theory: Learn about REST services and use Swagger OpenAPI to create a rest service that returns the CPU information about your computer

We will be starting the class with introducing you to REST services that provide a foundation for setting up services in the cloud and to interact with these services. As part of this class we will be revisiting the REST services and use them to deploy them on a cloud as well as develop our own AI based rest services in the second half of the class.

Focus on OpenAPI example posted in the NIST github repository

- *Project team:* Build a project team with no more than 3 people. There will not be an exception. You are allowed to work alone. Make sure your project team does the work together. E.g. YOu must not have 3 people on the team and the project could have been executed by a single person. In case of

more than one person the sum of the deliverables must be larger than what one team member can achieve. It is an advantage to work in a team as you can check each other.

If a team member does not contribute to the project, the team has the right to exclude the non working team member with consultation of the instructors. We will have a joint meeting with the team to identify the best path forward. Chose your team members wisely. Ideally you should make this decision in the first 3 weeks.

2.1.7.5 Week 4. Introduction to Scientific Writing

- Theory: Scientific writing with markdown and bibtex
- Practice: Contribute a significant chapter to the book (as a group)
- Practice: Project Report (as a group)
- Practice: Introduction to Emacs
- Practice: Introduction to jabref

See the separet ePub for more information: Link

Assignment Scientific Writing: Learn about markdown. See our class notes and internet resources. Note that we use pandoc markdown that may not render properly in github, especially when it comes to figure captions, references, and bibliography. (You have till the end of the month). Install and use jabref.

Report: Learn bibtex and create references in report.bib that you use in report.md. Make sure that you do only one report per team and update your README.yml file accordingly. Check in the Lab with the TAs if you have done it correct.

- *Project Idea due:* A one page formal document that summarizes the project. This is not a proposal. The workds *I* and *project, report* must not be used. It is essentially a snapshot of your final report. Discuss with the TAs in the Lab how to define a project.

  Github: make sure your team mates have access to your project directory.

2.1.7.6 Week 5 to 9. Introduction to Cloud Computing

- Theory:

  - Introduction - Part A
  - Introduction - Part B - Defining Clouds I
  - Introduction - Part C - Defining Clouds II
  - Introduction - Part D - Defining Clouds III
  - Introduction - Part E - Virtualization
  - Introduction - Part F - Technology Hypecycle I
  - Introduction - Part G - Technology Hypecycle II
  - Introduction - Part H - IaaS I
  - Introduction - Part I - IaaS II
  - Introduction - Part J - Cloud Software
  - Introduction - Part K - Applications I
  - Introduction - Part M - Applications III
  - Introduction - Part N - Parallelism
  - Introduction - Part O - Storage Released
  - Introduction - Part P - HPC in the Cloud Released
  - Introduction - Part Q - Analytics and Simulation Released
  - Introduction - Part R - Jobs Released
  - Introduction - Part S - The Future Released
  - Introduction - Part T - Security Released
  - Introduction - Part U - Fault Tolerance
- Practice: Manage virtual machines with Virtualbox
- Practice: Manage virtual machines with Cloudmesh v4
- Practice: Manage a container with Docker

- Theory: Containers

- *Week 5: Project Update due:* A two page formal document that summarizes the project. This is not a proposal. The words *I* and *project, report* must not be used. It is essentially a snapshot of your final report.

- *Week 7: Project Update due:* A multi-paragraph description about the data that you use for your project is to be added to your report. This includes details about the data. IN a documented program you show cases how you down load the data with python request in an automated fashion.

- *Week 8: Project Update Due:* Have a documented program ready that uses

a REST service to obtain data for your analysis. Identify how to do benchmarks and time the execution of your project. Add planed benchmarks to tour report. Do not use the word *plan* or *will* write it in such a form as if it were done. Instead put a **O** on benchmarks that you will that you wrok on

- *Week 9: Project update:* Study matplotlib and bokeah and identify how to visualize other aspects of your projects. Your are also allowed to use D3.js and add ons to it. You are **not allowed** to use tablaeu.

2.1.7.7 Week 10: Lecture Free Time

March 10 - 17 Lecture free time, no class support. A god week to work ahead on your project.

2.1.7.8 Week 11. Introduction to Cloud Platforms

We will introduce you to the concept of Map reduce. We will discuss systems such as Hadoop and Spark and how they differ. You will be deploying via a container hadoop on your machine and use it to gain hands on experience. We start with using cloudmesh on your computer to manage virtual machines that you may be able to use during your test developments.

Background about Hadoop, Spark and Twister

- Theory: Background to Cloudmesh
- Theory: Background to Hadoop
- Theory: Background to Spark

- Theory: Background to Twister

- *Week 11 Project update:* Identify analysis algorithms for your project and apply them. Experiment with what you can do with the data

2.1.7.9 Week 12 to 16. Review of AI for AI-Cloud Computing Integration

- Theory Introduction to basic AI

- Practice: Develop a non trivial AI REST service

See #sec:ai

- Overview of AI for this class
- Theory
- Unsupervised Learning
- Deep Learning

- Forecasting

- *Week 12: Project update:* Identify analysis algorithms for your project and apply them. Experiment with what you can do with the data

- *Week 13: Project update:* Identify analysis algorithms for your project and apply them. Experiment with what you can do with the data. Start benchmarks.

- *Week 14: Project update:* Focus on your project report and finalize it. The project report must include references in bibtex format. Double-check integration in proceedings.

- *Week 15: Apr 19 - Project due date.*

  As the Project will take time to grade all projects are due two weeks (yes, you read correctly) before the semester ends. The project will have the following artifacts:

  - completed project report
  - completed project code

  - completed instructions on you to replicate your project on someone else's computer or a cloud service

  - any other outstanding task.

- *Week 16: Apr 26*

  - Project improvement if needed (majority should be finished)

- Make sure your project report is showing up correctly in the proceedings

2.1.7.10 Cloud Edge Computing

If time allows we may in addition also cover.

- Theory: Raspberry PI as Platform

2.1.7.11 Alternative Projects

if you are interested the following could be chosen by you as project. Participation in these projects need to be approved by Dr. von Laszewski. The project starts in thsi case in week 2 or 3.

- Project (if elected): Document the build a 100 node Raspberry PI Cluster
- Project: Environmental Robot Boat

2.2 ASSIGNMENTS ☁

For more details see the course syllabus and overview pages. We give here just some summary.

2.2.1 Account Creation

As part of the class you will need a number of accounts

- piazza.com
- github.com

Optional accounts include (only apply for them if you know you need them. Note that applying for some accounts may take 1 - 2 weeks to complete, you should have identified before the middle of the semester if you need some of them.

- futuresystems.org (optional)
- chameleoncloud.org (optional)

- aws.com (optional)
- google.com (optional)
- azure.com (optional)
- watson from IBM (optional)
- google Iaas (optional)

In our piazza we have details how to submit them to us. We split the submission in multiple sub-assignments as the github.com and piazza.com are needed within the first week.

2.2.2 Sections, Chapters, Examples

As part of the class, we expect you to get familiar with topics related to intelligent systems engeneering. Thos that like to go for an A+ are also expected to contribute significantly to this document or have a truly outstanding project. This is done in Sections, Examples, and Chapters, or excelent Project reports and code.

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. Example sections that have been provided are for example the Lambda section in the python chapter
>
> Sample of student contributed sections include:

- [Project Natic](#)
- [Lambda Expressions](#)

◻ please fix links

Chapter:

> 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 **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. In addition it is expected if applicable 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 but sometimes we refer to it in our assignments as such. Chapters that focus on theory may not have an example and it can be substituted by a longer text.

A sample of a student contributed chapter is * [GraphQL](#).

Example:

An example is a document that showcases the use of a particular technology. Typically it is a console session or a program. Examples augment chapters and Sections.

⚠️ *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 emphasize here is that we do not decide them for you, but we identify them with you.*

Sample Topics that could form a section or chapter are clearly marked with a ⍰. There are plenty in the handbook, but you are welcome to define your own contributions. Discuss them with us in the online hours.

A list of topics identified by students is maintained in a spreadsheet.

See https://piazza.com/class/jgxybbf5rnx5qd?cid=201 for details.

You are expected to signup in this spreadsheet. THis is done to ab=void overlap and foster uniqueness of the assignment for sections and chapters.

2.2.3 Project

Project:

> We refer with the term project to the major activity that you chose as part of your class. The default case is an implementation project that requires a *project report* and project code.

License:

> All projects are developed under an open source license such as Apache 2.0 License. You will be required to add a LICENCE.txt file and if you use other software identify how it can be reused in your project. If your project uses different licenses, please add in a README.md file which packages are used and which license these packages have.

Project Report:

> A project report is an enhanced topic paper that includes not just the analysis of a topic, but an actual code, with **benchmark** and demonstrated application use. Obviously it is longer than a term paper and includes descriptions about reproducibility of the application. A README.md is provided that describes how others can reproduce your project and run it. Remember tables and figures do not count towards the paper length. The following length is required:

- 4 pages, one student in the project
- 6 pages, two students in the project
- 8 pages, three students in the project

We estimate that a single page is between 1000-1200 words. Please note that for 2018 the format will be markdown, so the word count will be used instead. How to use figures is explained in the Notation of the handbook. We use bibtex for bibliographies. Please be reminded that images and tables as well as code is excluded from the page length. Make sure that your text is mostly developed by midterm time.

Project Code:

This is the **documented** and **reproducible** code and scripts that allows a TA do replicate the project. In case you use images they must be created from scratch locally and may not be uploaded to services such as dockerhub. You can however reuse vendor uploaded images such as from ubuntu or centos. All code, scripts, and documentation must be uploaded to github.com under the class specific github directory.

Data:

Data is to be hosted on IUs google drive if needed. If you have larger data, it should be downloaded from the internet. It is in your responsibility to develop a download program. The data **must** not be stored in github. You will be expected to write a python program that downloads the data.

Work Breakdown:

This is an appendix to the document that describes in detail who did what in the project. This section comes in a new page after the references. It does not count towards the page length of the document. It also includes explicit URLs to the git history that documents the statistics to demonstrate not only one student has worked on the project. If you can not provide such a statistic or all check-ins have been made by a single student, the project has shown that they have not properly used git. Thus points will be deducted from the project. Furthermore, if we detect that a student has not contributed to a project we may invite the student to give a detailed presentation of the project.

Bibliography:

All bibliography has to be provided in a jabref/bibtex file. There is **NO EXCEPTION** to this rule. Please be advised doing references right takes some time so you want to do this early. Please note that exports of Endnote or other bibliography management tools do not lead to properly formatted bibtex files, despite they claiming to do so. You will have to clean them up and we recommend to do it the other way around. Manage your bibliography with jabref. Make sure labels only include chracters from [a-zA-Z0-9-]. Use dashes and not underscore or : in the label.

2.2.3.1 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 importnat 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 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.

You have plenty of time to make execute a wonderful project.

The deliverables include but need to be updated according to your specific project, for example if you do Edge Computing some deliverabl;es will be different:

- Provide benchmarks.

- Take results in two different cloud services and your local PC (ex: Chameleon Cloud, echo kubernetes). Make sure your system can be created and deployed based on your documentation.

- Each team member must provide a benchmark on their computer and a cloud IaaS, where the cloud is different from each team member.

- Create a Makefile with the tags deploy, run, kill, view, clean that deploys your environment, runs application, kills it, views the result and cleans up after wards. You are allowed to have different makefiles for the different clouds and different directories. Keep the code and directory structure clean and document how to reproduce your results.

- For python use a requirements.txt file also

- For docker use a Dockerfile also

- Write a report that includes the following sections

- Abstract
- Introduction
- 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 incomplte, instead you can use an optional "Limitations" section.

2.2.3.2 Project Topic

As part of this class you will be developing a OpenAPI based Artificial Intelligence REST service and demonstrate its use. YOu will be developing a documentation and a report that showcases the use of the service. The OpenAPI service must be non trivial, e.g. you shoudl show upload of data, sbmission of parameters including the function to be executed, potential development of a GUI for the service.

We will work with you to solidify the project throughout the semester.

2.2.4 Alternate Project: Virtual Cluster

All students can contribute to the creation of the Virtual Cluster code that we will be using throughout the class to improve and interface with cloud and container frameworks. This project is typically done in a graduate class, but interested undergraduates can contribute also. Those that like to contribute must have significant programming experience in either Python or Javascript. This project could replace the regilar AI REST service project. A weekly meeting and demonstrated progress has to be shown to Gregor von Laszewski.

- https://github.com/cloudmesh-community/cm

The residential students have been assigned this task, but online students can join and contribute.

2.2.5 Alternative Project: 100 node Raspberry Pi cluster

In this project you will be developing a 100 node Raspberry PI cluster. THis includes putting the hardware together, and developing software that allows to uses all 100 nodes as a cluster. Software is to be use to make management easiy. It is not sufficient to just install the software but to develop a framework that allows us to easily share this resource with other users.

A documentation has to be written for this project so others can replicate your cluster build. A good start for this is to look at our `cm-burn` command that creates Raspberry PI OS based on manipulation of the file system

- https://github.com/cloudmesh/cm-burn
- https://github.com/cloudmesh-community/cm

Substantial contributions are expected beyond the hardware build. We also like to design a case with a Laser cutter for the 100 nodes. Building the cluster would take place in MESH and transportation to and from it is provided by the university. You will be able to work in an office there to put the cluster together. A weekly meeting with Gregor von Laszewski or the TAs is needed to showcase progress.

2.2.6 Submission of sections and chapters and projects

Sections and subsections are to be added to the `book` github repo. Do a pull request. The headline of the section needs to be marked with a 🔧 if you still work on it, marked with a ☺ if you want it to be graded. and have all hids for people that contribute to that section.

In addition, simply add them to your README.yml file in your github repo. Add the following to it (I am using a18-516-18 as example).

Please look at https://github.com/cloudmesh-community/fa18-516-18 and https://raw.githubusercontent.com/cloudmesh-community/fa18-523-62/master/README.yml for an examples. Please note that in case you work in a group the code and report is supposed to be only stored in the first hid mentioned in the group field. If you store it in multiple directories your project will be rejected.

```
section:
    - title: title of the section 1
      url: https://github.com/cloudmesh-community/book/chapters/...
    - title: title of the section 2
      url: https://github.com/cloudmesh-community/book/chapters/...
    - title: title of the section 3
      url: https://github.com/cloudmesh-community/book/chapters/...
chapter:
    - title: title of the chapter
      url: https://github.com/cloudmesh-community/fa18-516-18/blob/master/chapter/whatever.md
      group: fa18-523-62 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 give point deductions.

3 PYTHON

3.1 INTRODUCTION TO PYTHON

Learning Objectives

- Learn quickly Python under the assumption you know a programming language
- Work with modules
- Understand docopts and cmd
- Contuct some python examples to refresh your python knpwledge
- Learn about the `map` function in Python
- Learn how to start subprocesses and rederect 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 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
- have an overview of the standard library
- install Python libraries using pyenv for multipython interpreter development.

E doe 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 2.7.16 or 3.7.4
- Familiarity with command line usage
- A text editor such as [PyCharm](), emacs, vi or others. You should identity which works best for you and set it up.

3.1.1 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/

3.2 PYTHON 3.7.4 INSTALLATION ☁

🎓 Learning Objectives

- Learn how to install python.
- Find additional information about Python.
- Make sure your Computer supports Python.

In this setion we explain how to install python 3.7.4 on a computer. Likely much of the code will work with earlier versions, but we do the development in Python on the newest version of python available at https://www.python.org/downloads .

3.2.1 Hardware

Python does not require any special hardware. We have installed Python not only on PC's and Laptops, but also on Raspberry PI's and Lego Mindstorms.

However, there are some things to consider. If you use many programs on your desktop and run them all at the same time you will find that in up-to-date operating systems you will find your self quickly out of memmory. This is especially true if you use editors such as PyCharm which we highly recommend. Furthermore, as you likely have lots of disk access, make sure to use a fast HDD or better an SSD.

A typical modern developer PC or Laptop has *16GB RAM* and an *SSD*. You can certainly do python on a $35 Rapbperry PI, but you probably will not be able to run PyCharm. There are many alternative editors with less Memory footprint avialable.

3.2.2 Prerequisits Ubuntu 19.04

Python 3.7 is installed in ubuntu 19.04. Therefore, it already fulfills the prerequisits. However we recommend that you update to the newest version of python and pip. However we recommend that you update the the newest version of python. Please visit: https://www.python.org/downloads

3.2.3 Prerequisits macOS

3.2.3.1 Installation from Apple App Store

You want a number of useful tool on your macOS. They are not installed by default, but are available via Xcode. First you need to install xcode from

- https://apps.apple.com/us/app/xcode/id497799835

Next you need to install macOS xcode command line tools:

```
$ xcode-select --install
```

3.2.3.2 Installation from python.org

The easiest instalation of Python for cloudmesh is to use the instaltion from https://www.python.org/downloads. Please, visit the page and follow the instructions. After this install you have python3 avalable from the commandline

3.2.3.3 Installation from Hoembrew

An alternative instalation is provided from Homebrew. To use this install method, you need to install Homebrew first. Start the process by installing the python 3 using `homebrew`. Install `homebrew` using the instruction in their web page:

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Then you should be able to install Python 3.7.4 using:

```
$ brew install python
```

3.2.4 Prerequisits Ubuntu 18.04

We recommend you update your ubuntu version to 19.04 and follow the instructions for that version instead, as it is significantly easier. If you however are not able to do so, the following instructions may be helpful.

We first need to make sure that the correct version of the Python3 is installed. The default version of Python on Ubuntu 18.04 is 3.6. You can get the version with:

```
$ python3 --version
```

If the version is not 3.7.4 or newer, you can update it as follows:

```
$ sudo apt-get update
$ sudo apt install software-properties-common
$ sudo add-apt-repository ppa:deadsnakes/ppa
$ sudo apt-get install python3.7 python3-dev python3.7-dev
```

You can then check the installed version using `python3.7 --version` which should be `3.7.4`.

Now we will create a new virtual environment:

```
$ python3.7 -m venv --without-pip ~/ENV3
```

The edit the `~/.bashrc` file and add the following line at the end:

```
alias ENV3="source ~/ENV3/bin/activate"
ENV3
```

now activate the virtual environment using:

```
$ source ~/.bashrc
```

now you can install the pip for the virtual environment without conflicting with the native pip:

```
$ curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
$ python get-pip.py
$ rm get-pip.py
```

3.2.5 Prerequisite Windows 10

Python 3.7 can be installed on Windows 10 using:
https://www.python.org/downloads

For 3.7.4 can go to the download page and download one of the different files for Windows.

Let us assume you choe the Web based installer, than you click on the file in the edge browser (make sure the account you use has administrative priviledges). Follow the instructions that the installer gives. Important is that you select at one point "[x] Add to Path". There will be an empty checkmark about this that you will click on.

Once it is installed. chose a terminal and execute

```
python --version
```

However, if you have installed conda for some reason you need to read up on how to install 3.7.4 python in conda or identify how to run conda and python.org at the same time. We see often others giving the wrong installation instructions.

An alternative is to use python from within the Linux Subsystem. But that has some limitations and you will need to explore how to exxess the file system in the subssytem to have a smooth integration between your Windows host so you can for example use PyCharm.

3.2.5.1 Linux Subsystem Install

To activate the Linux Subsystem, please follow the instructions at

* https://docs.microsoft.com/en-us/windows/wsl/install-win10

A suitable distribution would be

* https://www.microsoft.com/en-us/p/ubuntu-1804-lts/9n9tngvndl3q?activetab=pivot:overviewtab

However as it uses an older version of python you will ahve to update it.

3.2.6 Prerequisit venv

This step is highly recommend if you have not yet already installed a `venv` for python to make sure you are not interfering with your system python. Not using a venv could have catastrophic consequences and a destruction of your operating system tools if they realy on Python. The use of venv is simple. For our purposes we assume that you use the directory:

```
~/ENV3
```

Follow these steps first:

First cd to your home directory. Then execute

```
$ python3 -m venv  ~/ENV3
$ source ~/ENV3/bin/activate
```

You can add at the end of your .bashrc (ubuntu) or .bash_profile (macOS) file the line

```
$ source ~/ENV3/bin/activate
```

so the environment is always loaded. Now you are ready to install cloudmesh.

Check if you have the right version of python installed with

```
$ python --version
```

To make sure you have an up to date version of pip issue the command

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

3.2.7 Install Python 3.7 via Anaconda

3.2.7.1 Download conda installer

Miniconda is recommended here. Download an installer for Windows, macOS, and Linux from this page: https://docs.conda.io/en/latest/miniconda.html

3.2.7.2 Install conda

Follow instructions to install conda for your operating systems:

- Windows. https://conda.io/projects/conda/en/latest/user-guide/install/windows.html
- macOS. https://conda.io/projects/conda/en/latest/user-guide/install/macos.html
- Linux. https://conda.io/projects/conda/en/latest/user-guide/install/linux.html

3.2.7.3 Install Python 3.7.4 via conda

```
$ cd ~
$ conda create -n ENV3 python=3.7.4
$ conda activate ENV3
$ conda install -c anaconda pip
$ conda deactivate ENV3
```

It is very important to make sure you have a newer version of pip installed. After you installed and created the ENV3 you need to activate it. This can be done

with

```
$ conda activate ENV3
```

If you like to activate it when you start a new terminal, please add this line to your `.bashrc` or `.bash_profile`

If you use zsh please add it to `.zprofile` instead.

3.3 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**: **R**ead-**E**valuate-**P**rint-**L**oop.

3.3.1 REPL (Read Eval Print Loop)

There are many different types beyond what we have seen so far, such as **dictionaries**s, **list**s, **set**s. 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.

3.3.2 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.

3.3.3 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
```

3.4 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.

3.4.1 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](#)

3.4.2 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 the next 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.

3.5 LANGUAGE ☁

3.5.1 Statements and Strings

Let us explore the syntax of Python while starting with a print statement

```
print("Hello world from Python!")
```

This will print on the terminal

```
Hello world from Python!
```

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.

3.5.2 Comments

Comments in python are followed by a `#`:

```
# This is a comment
```

3.5.3 Variables

You can store data into a **variable** to access it later. For instance:

```
hello = 'Hello world from Python!'
print(hello)
```

This will print again

```
Hello world from Python!
```

3.5.4 Data Types

3.5.4.1 Booleans

A **boolean** is a value that can have the values `True` or `False`. You can combine booleans with **boolean operators** such as `and` 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
```

3.5.4.2 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 yth power.

You can combine **float**s and **int**s:

```
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
```

3.5.5 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.

3.5.5.1 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.

3.5.5.2 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
```

3.5.6 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': [1, 0, 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]
```

3.5.7 Control Statements

3.5.7.1 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.

```python
x = int(input("Guess x:"))
if x == 4:
    print('Correct!')
elif abs(4 - x) == 1:
    print('Wrong, but close!')
else:
    print('Wrong, way off!')
```

3.5.7.2 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:

```python
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:

```python
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:

```python
for i in range(10):
    print(i + 1)
```

We can also nest loops inside each other:

```python
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)

3.5.8 Datatypes

3.5.8.1 Lists

see: 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]
```

3.5.8.2 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](#).

3.5.8.3 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 us 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.

3.5.8.4 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
```

3.5.8.5 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*.

3.5.8.6 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:

```python
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:

```python
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](#).

3.5.9 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:

```python
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.

```python
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)
```

3.5.10 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:

```python
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](#).

3.5.11 Modules

Now write this simple program and save it:

```
print("Hello world!")
```

As a check, make sure the file contains the expected contents on the command line:

```
$ cat hello.py
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 **module**s. 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 us 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 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.

3.5.12 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:

```python
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:

```python
power2 = lambda x: x ** 2
```

The `power2` function defined in the expression, is equivalent to the following definition:

```python
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 `filer` 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`.

```python
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.

```python
list_a = [1, 2, 3, 4, 5]
filter_obj = filter(lambda x: x % 2 == 0, list_a)
# Convert the filer 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:

```python
lambda arguments : expression
```

For an example: a function in python

```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:

```python
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.

```python
(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.

3.5.12.1 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]
```

3.5.12.2 dictionary

Now, lets see how we can interate 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 next:

```
map_output = map(lambda x: x*2, [1, 2, 3, 4])
print(map_output)
# Output: map object: <map object at 0x04D6BAB0>
list_map_output = list(map_output)
print(list_map_output) # Output: [2, 4, 6, 8]
```

3.5.13 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 `log2` of numbers:

```python
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 `log2()` of 1, 2, 3, 4 respectively.

3.5.14 Generators

Before we go to Generators, please understand Iterators. Generators are also Iterators but they can only be interated over once. Thats 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.

3.5.14.1 Generators with function

For example: we have a function named as multiplyBy10 which prints all the input numbers multiplied by 10.

```python
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.

```python
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 meanns Generators has exausted its limit and it has no 6th element to print.

```python
print next(new_numbers)  #Output: 1
```

3.5.14.2 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
```

3.5.14.3 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.

3.5.14.4 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.

```python
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 (f'Memory (Before): {memory}Mb')

def people_list(people):
    result = []
    for i in range(people):
        person = {
                'id' : i,
                'name' : random.choice(names),
                'major' : randon.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 (f'Memory (After): {memory}Mb')
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.

```python
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 (f'Memory (Before): {memory}Mb')
def people_generator(people):
    for i in xrange(people):
        person = {
            'id' : i,
            'name' : random.choice(names),
            'major' : randon.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 (f'Memory (After): {memory}Mb')
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 alomost 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.

3.6 LIBRARIES

3.6.1 Python Modules ☁

Often you may need functionality that is not present in Python's standard library. In this case you have two option:

- 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 if virtual environments using the pyenv or virtualenv command, and installing and uninstalling PyPi packages using pip.

3.6.1.1 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 python and you use pyenv, than you can update pip with

```
pip install -U pip
```

without interfering with a potential system wide installed version of p ip that

may be needed by the system default version of python. See the section about pyenv for more details

3.6.1.2 Using pip to Install Packages

Let us 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.

3.6.1.3 GUI

3.6.1.3.1 GUIZero

Install guizero with the following command:

```
sudo pip install guizero
```

For a comprehensive tutorial on guizero, [click here](#).

3.6.1.3.2 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 fine here

- https://github.com/cloudmesh/cloudmesh.robot/tree/master/projects/kivy

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
```

3.6.1.4 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.

3.6.1.5 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.

3.6.1.6 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

3.6.1.7 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.

3.6.1.8 Ecosystem

3.6.1.8.1 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.

3.6.1.8.2 Alternative Installations

The basic installation of python is provided by 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 form 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 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 form pypi which is standard and should be supported by the frameworks. We have not tested cloudmesh on any other python version then 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 an 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 form github
- deploy run and document cloudmesh on canopy, develop script to generate a conda package form github
- deploy run and document cloudmesh on ironpython
- other documentation that would be useful

3.6.1.9 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/handout.html
- http://www.youtube.com/watch?v=0vJJlVBVTFg
- http://www.korokithakis.net/tutorials/python/
- http://www.afterhoursprogramming.com/tutorial/Python/Introduction/
- http://www.greenteapress.com/thinkpython/thinkCSpy.pdf
- https://docs.python.org/3.3/tutorial/modules.html
- https://www.learnpython.org/en/Modules/_and/_Packages
- https://docs.python.org/2/library/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

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.

3.6.1.9.1 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.

3.6.1.10 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.*

3.6.2 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.

3.6.2.1 Formats

3.6.2.1.1 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.

```python
import pickle

flavor = {
    "small": 100,
    "medium": 1000,
    "large": 10000
    }

pickle.dump( flavor, open( "data.p", "wb" ) )
```

To read it back in use

```python
flavor = pickle.load( open( "data.p", "rb" ) )
```

3.6.2.1.2 Text Files

To read text files into a variable called content you can use

```python
content = open('filename.txt', 'r').read()
```

You can also use the following code while using the convenient `with` statement

```python
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 cam 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)
```

3.6.2.1.3 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 swill split at every comma, including those contained in quotes. So this method although looking originally convenient has limitations.

3.6.2.1.4 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')
```

3.6.2.1.5 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

```python
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

```python
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.

3.6.2.1.6 JSON

```python
import json
with open('strings.json') as f:
    content = json.load(f)
```

3.6.2.1.7 XML

XML format is extensively used to transport data across the web. It has a hierarchical data format, and can be represented in the form of a tree.

A Sample XML data looks like:

```xml
<data>
    <items>
        <item name="item-1"></item>
        <item name="item-2"></item>
        <item name="item-3"></item>
    </items>
</data>
```

Python provides the ElementTree XML API to parse and create XML data.

Importing XML data from a file:

```python
import xml.etree.ElementTree as ET
tree = ET.parse('data.xml')
root = tree.getroot()
```

Reading XML data from a string directly:

```python
root = ET.fromstring(data_as_string)
```

Iterating over child nodes in a root:

```
for child in root:
    print(child.tag, child.attrib)
```

Modifying XML data using ElementTree:

- Modifying text within a tag of an element using .text method:

```
tag.text = new_data
tree.write('output.xml')
```

- Adding/modifying an attribute using .set() method:

```
tag.set('key', 'value')
tree.write('output.xml')
```

Other Python modules used for parsing XML data include

- minidom: https://docs.python.org/3/library/xml.dom.minidom.html
- BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/

3.6.2.1.8 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
```

3.6.2.1.9 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 than 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.

3.6.2.1.10 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

Beautiful Soup is a python library to parse, process and edit HTML documents.

To install Beautiful Soup, use `pip` command as follows:

```
$ pip install beautifulsoup4
```

In order to process HTML documents, a parser is required. Beautiful Soup

supports the HTML parser included in Python's standard library, but it also supports a number of third-party Python parsers like the `lxml` parser which is commonly used [1].

Following command can be used to install `lxml` parser

```
$ pip install lxml
```

To begin with, we import the package and instantiate an object as follows for a html document `html_handle`:

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_handle, `lxml`)
```

Now, we will discuss a few functions, attributes and methods of Beautiful Soup.

prettify function

`prettify()` method will turn a Beautiful Soup parse tree into a nicely formatted Unicode string, with a separate line for each HTML/XML tag and string. It is analgous to `pprint()` function. The object created above can be viewed by printing the prettfied version of the document as follows:

```
print(soup.prettify())
```

tag Object

A `tag` object refers to tags in the HTML document. It is possible to go down to the inner levels of the DOM tree. To access a tag `div` under the tag `body`, it can be done as follows:

```
body_div = soup.body.div
print(body_div.prettify())
```

The `attrs` attribute of the tag object returns a dictionary of all the defined attributes of the HTML tag as keys.

has_attr() method

To check if a `tag` object has a specific attribute, `has_attr()` method can be used.

```
if body_div.has_attr('p'):
    print('The value of \'p\' attribute is:', body_div['p'])
```

tag object attributes

- `name` - This attribute returns the name of the tag selected.
- `attrs` - This attribute returns a dictionary of all the defined attributes of the HTML tag as keys.
- `contents` - This attribute returns a list of contents enclosed within the HTML tag
- `string` - This attribute which returns the text enclosed within the HTML tag. This returns `None` if there are multiple children
- `strings` - This overcomes the limitation of `string` and returns a generator of all strings enclosed within the given tag

Following code showcases usage of the above discussed attributes:

```
body_tag = soup.body

print("Name of the tag:', body_tag.name)

attrs = body_tag.attrs
print('The attributes defined for body tag are:', attrs)

print('The contents of \'body\' tag are:\n', body_tag.contents)

print('The string value enclosed in \'body\' tag is:', body_tag.string)

for s in body_tag.strings:
    print(repr(s))
```

Searching the Tree

- `find()` function takes a filter expression as argument and returns the first match found
- `findall()` function returns a list of all the matching elements

```
search_elem = soup.find('a')
print(search_elem.prettify())

search_elems = soup.find_all("a", class_="sample")
pprint(search_elems)
```

- `select()` function can be used to search the tree using CSS selectors

```
# Select `a` tag with class `sample`
a_tag_elems = soup.select('a.sample')
print(a_tag_elems)
```

3.6.2.1.11 ConfigParser

🄾 TODO: Students can contribute a section

- https://pymotw.com/2/ConfigParser/

3.6.2.1.12 ConfigDict

- https://github.com/cloudmesh/cloudmesh-common/blob/master/cloudmesh/common/ConfigDict.py

3.6.2.2 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 se 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.

```sh
#! /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.

```python
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`.

3.6.2.3 Database Access

🅞 TODO: Students: define conventional database access section

see: https://www.tutorialspoint.com/python/python_database_access.htm

3.6.2.4 SQLite

🅞 TODO: Students can contribute to this section

https://www.sqlite.org/index.html

https://docs.python.org/3/library/sqlite3.html

3.6.2.4.1 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*

3.6.3 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 next.

```
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 out 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:

```
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, next 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 next using data from [T:fast-cars].

```python
import matplotlib.pyplot as plt

x = [' Toyota Prius',
     'Tesla Roadster ',
     ' Bugatti Veyron',
     ' Honda Civic ',
     ' Lamborghini Aventador ']
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.

```python
print(plt.style.available)
```

An example of using a predefined style is shown next.

```python
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.

```python
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)
```

3.6.4 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 apprach 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/options_example.py

A sample program of using doc opts for our purposes loks as follows

```python
"""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.

3.6.5 Cloudmesh Command Shell ☁

3.6.5.1 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.

3.6.5.1.1 Resources

The source code for cmd5 is located in github:

- https://github.com/cloudmesh/cmd5

We have discussed in Section **¿sec:cloudmesh-cms-install?** how to install cloudmesh as developer and have access to the source code in a directory called `cm`. As you read this document we assume you are a developer and can skip the next section.

3.6.5.1.2 Installation from source

WARNING: DO NOT EXECUTE THIS IF YOU ARE A DEVELOPER OR YOUR ENVIRONMENT WILL NOT PROPERLY WORK.

However, if you are a user of cloudmesh you can install it with

```
$ pip install cloudmesh-cmd5
```

3.6.5.1.3 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:

```
+-----------------------------------------------------+
|   ____ _   _                   _               _    |
|  / ___| | ___  _   _  __| | _ __ ___   ___  ___| |__  |
| | |   | |/ _ \| | | |/ _` | '_ ` _ \ / _ \/ __| '_ \ |
| | |___| | (_) | |_| | (_| | | | | | |  __/\__ \ | | | |
|  \____|_|\___/ \__,_|\__,_|_| |_| |_|\___||___/_| |_| |
+-----------------------------------------------------+
|                 Cloudmesh CMD5 Shell                |
+-----------------------------------------------------+

cms>
```

To see the list of commands you can say:

```
cms> help
```

To see the manual page for a specific command, please use:

```
help COMMANDNAME
```

3.6.5.1.4 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* cloudmesh command (we suggest you use a different name than `gregor` maybe your firstname):

```
$ cms sys command generate gregor
```

It will download a template from cloudmesh called `cloudmesh-bar` and generate a new directory `cloudmesh-gregor` 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-gregor
$ 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 use 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:

```
cloudmsesh/mycommand/command/gregor.py
```

Now you can go ahead and modify your command in that directory. It will look similar to (if you used the command name `gregor`):

```python
from __future__ import print_function
from cloudmesh.shell.command import command
from cloudmesh.shell.command import PluginCommand

class GregorCommand(PluginCommand):

    @command
    def do_gregor(self, args, arguments):
        """
        ::
          Usage:
                gregor -f FILE
                gregor list
          This command does some useful things.
          Arguments:
              FILE   a file name
          Options:
              -f       specify the file
        """
        print(arguments)
        if arguments.FILE:
          print("You have used file: ", arguments.FILE)
```

```
        return ""
```

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.

3.6.5.1.5 Bug: Quotes

We have one bug in cmd5 that relates to the use of quotes on the commandline

For example you need to say

```
$ cms gregor -f \"file name with spaces\"
```

If you like to help us fix this that would be great. it requires the use of shlex. Unfortuantly we did not yet time to fix this "feature".

3.6.6 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.

3.6.6.1 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.

```python
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.

3.6.6.2 A More Involved Example

Let us look at a little more involved example. Save the following code in a file called calculator.py.

```python
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.

3.6.6.3 Help Messages

Notice that all commands presently show up as undocumented. To remedy this, we can define help_ methods for each command:

```python
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)
```

```python
    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
```

3.6.6.4 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](#)

3.6.7 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.

3.6.7.1 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).

3.6.7.2 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:

-

Note that building from source can take a long time and may not be feasible for deploying to limited platforms such as Raspberry Pi.

3.6.7.3 A Simple Example

In this example, an image is loaded. A simple processing is performed, and the result is written to a new image.

3.6.7.3.1 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

3.6.7.3.2 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 1](#)
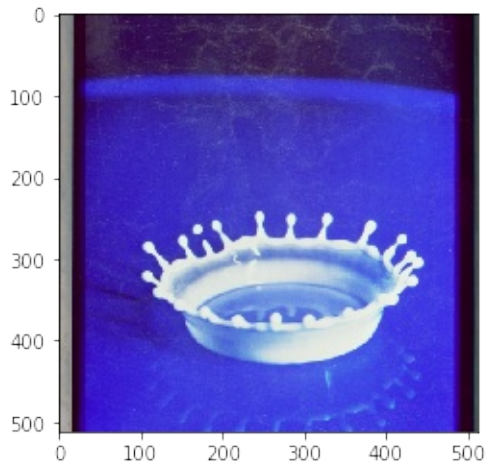
Figure 1: Image display

3.6.7.3.3 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 2](#)
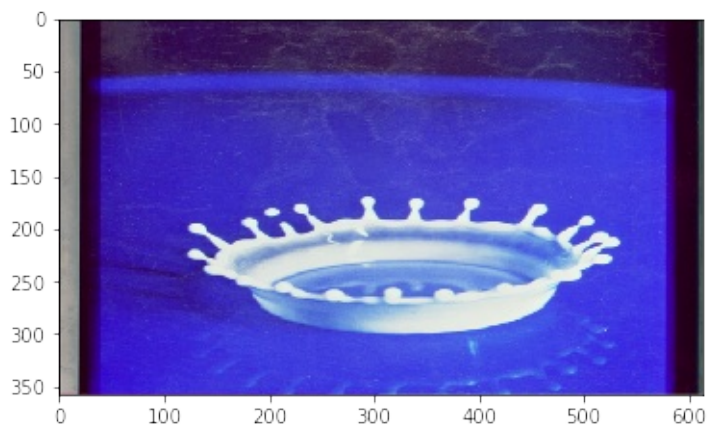
Figure 2: Scaling and rotation

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 3](#)
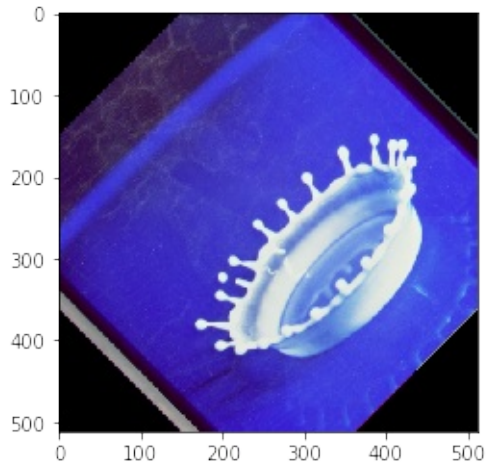
Figure 3: image

3.6.7.3.4 Gray-scaling

```
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img2, cmap='gray')
```

which results in +[Figure 4](#)
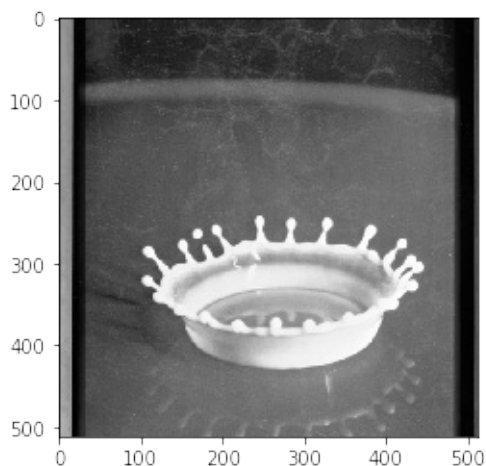
Figure 4: Gray sacling

3.6.7.3.5 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 5](#)
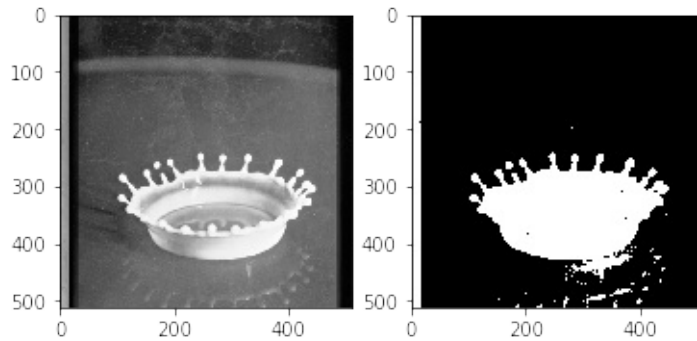
Figure 5: Image Thresholding

3.6.7.3.6 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 6](#)
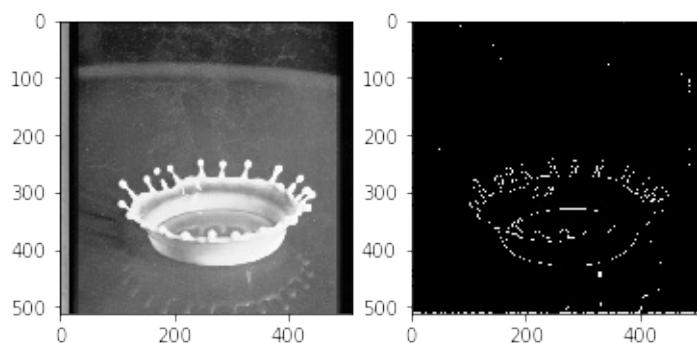
Figure 6: Edge detection

3.6.7.4 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.

3.6.8 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. Than 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 7
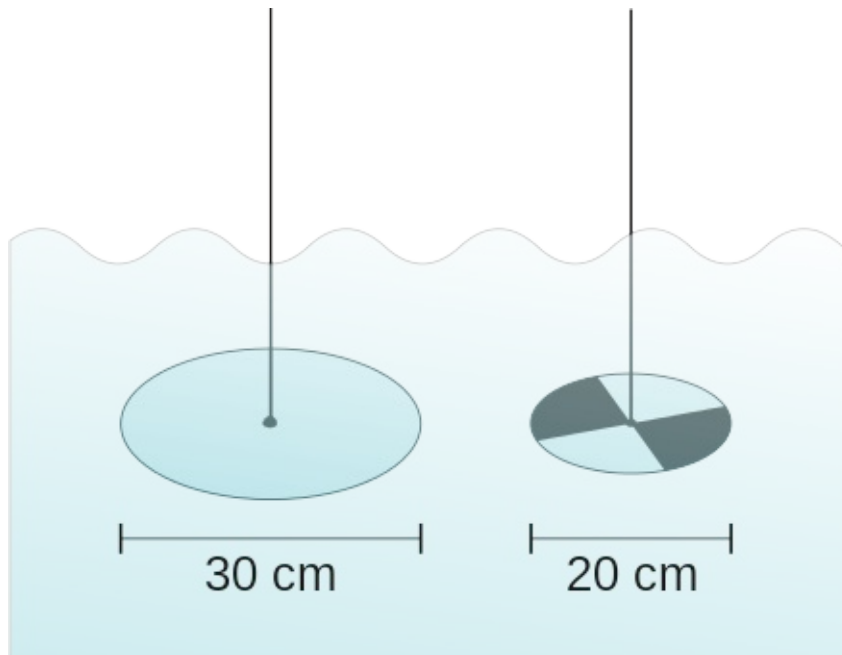
Figure 7: Secchi disk types. A marine style on the left and the freshwater version on the right wikipedia.

More information about Secchi Disk can be found at:

- https://en.wikipedia.org/wiki/Secchi/_disk

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.

3.6.8.1 Setup for OSX

First lest 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, pleas elt us know. However we do not want to install it via Anaconda out of the obvious reason that anaconda installs to 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
```

3.6.8.2 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 8](#)

3.6.8.3 Step 2: Analyse the images from the Video

For now we just selected 4 images from the video

```python
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()
```
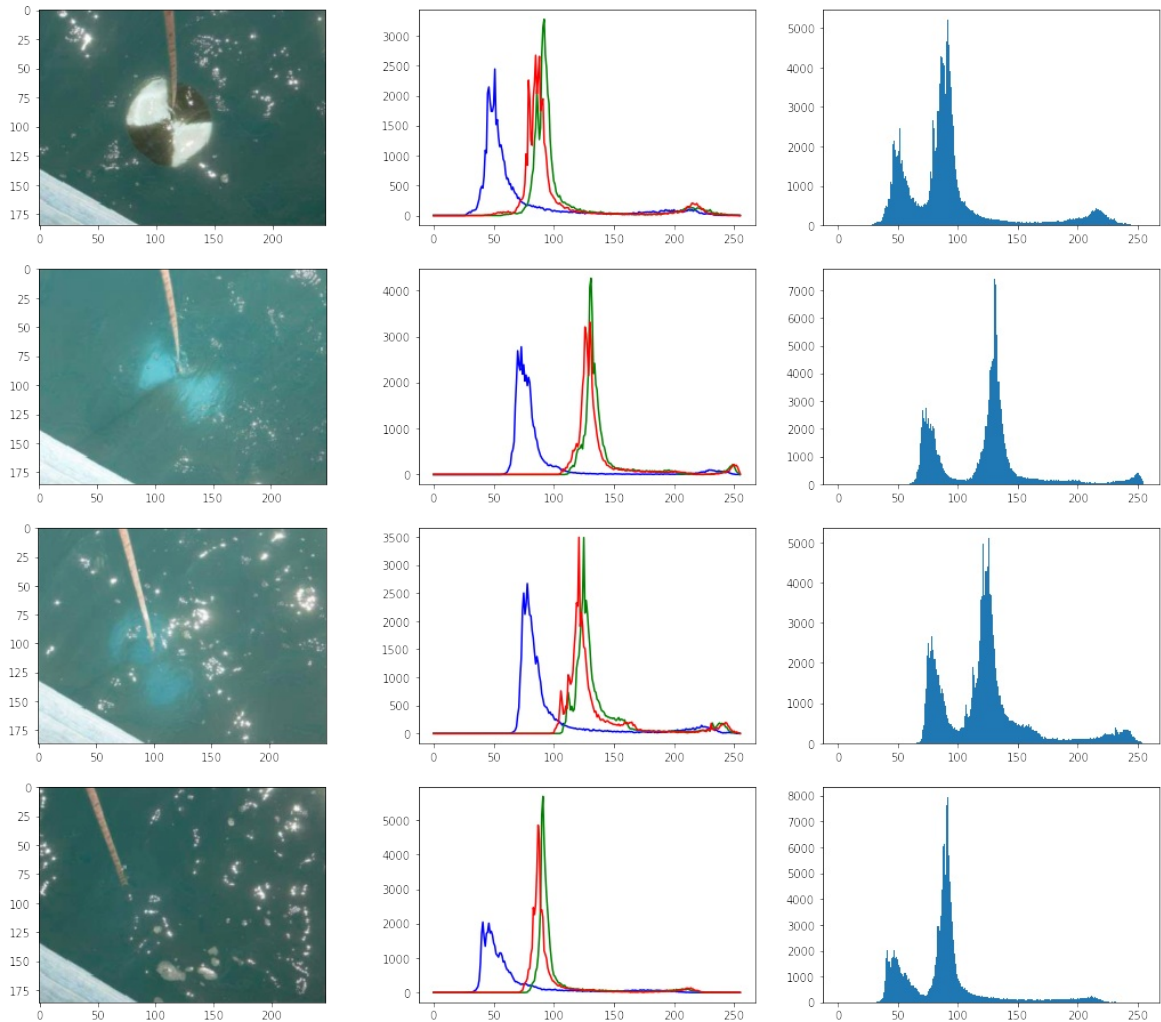
Figure 8: Histogram

3.6.8.3.1 Image Thresholding

See [Figure 9](#), [Figure 10](#), [Figure 11](#), [Figure 12](#)

```
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)
```
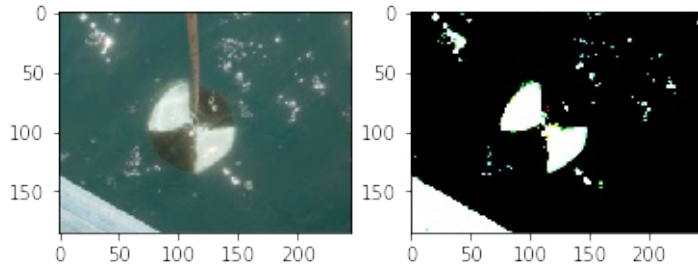
Figure 9: Threshold 1

```
threshold(img2)
```
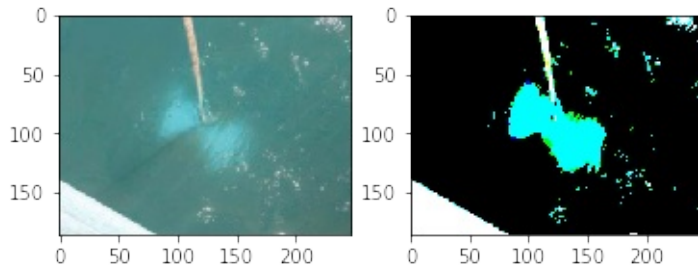

Figure 10: Threshold 2

```
threshold(img3)
```
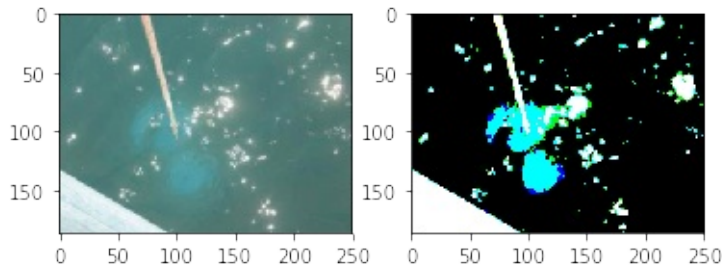

Figure 11: Threshold 3

```
threshold(img4)
```
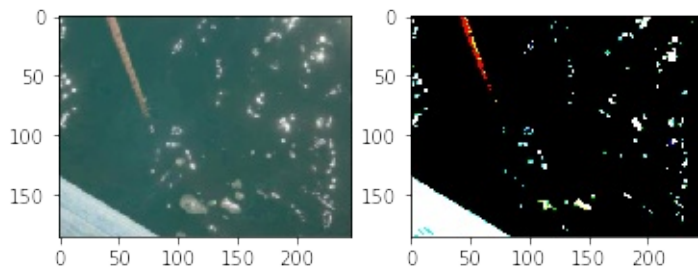

Figure 12: Threshold 4

3.6.8.3.2 Edge Detection

See [Figure 13](#), [Figure 14](#), [Figure 15](#), [Figure 16](#), [Figure 17](#). 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)
```
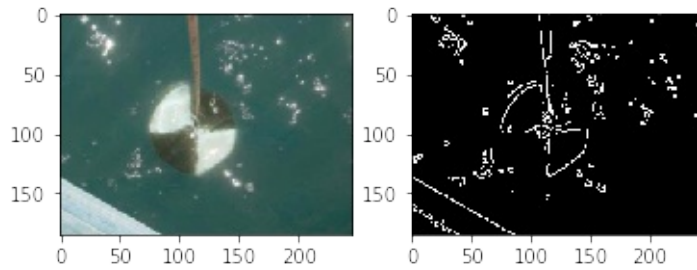

Figure 13: Edge Detection 1

```
find_edge(img2)
```
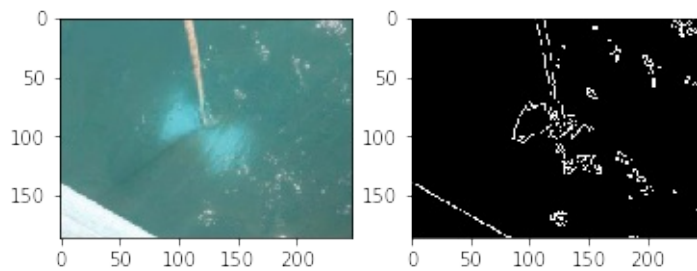

Figure 14: Edge Detection 2

```
find_edge(img3)
```
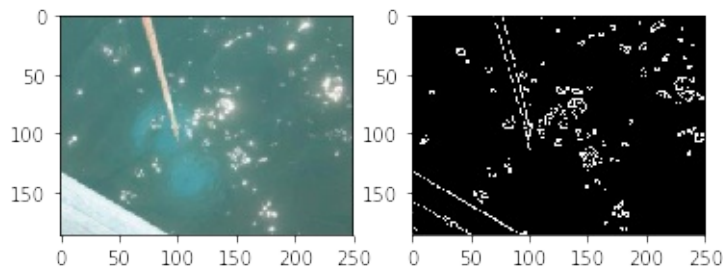

Figure 15: Edge Detection 3

```
find_edge(img4)
```
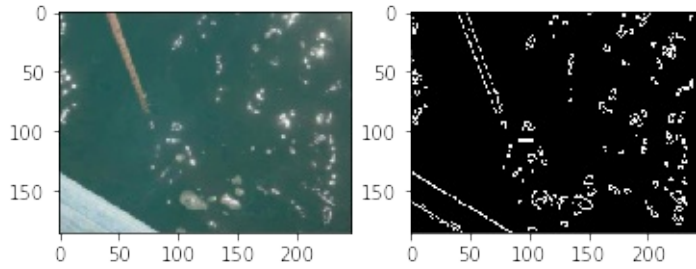
Figure 16: Edge Detection 4

3.6.8.3.3 Black and white

```
bw1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
plt.imshow(bw1, cmap='gray')
```
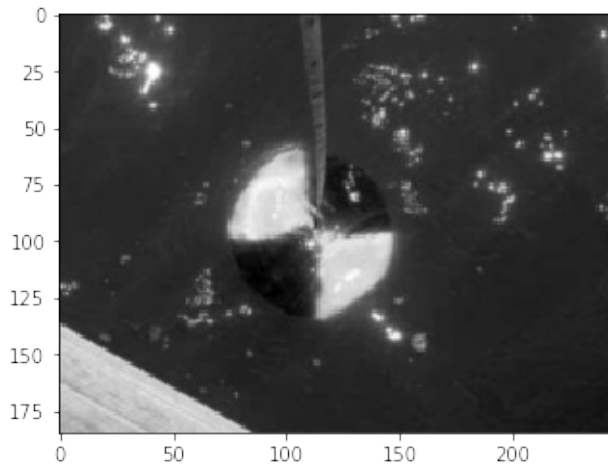

Figure 17: Back White conversion

3.7 DATA

3.7.1 Data Formats ☁

3.7.1.1 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 tat YAM in contrast to XML removes the tags while replacing them with indentation. This has naturally the advantage that it is mor 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 read 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.

```python
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/

3.7.1.2 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/

3.7.1.3 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

3.7.2 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 [2]. 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 [3]. 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 [3]. 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.

3.7.2.1 Cloudmesh MongoDB Usage Quickstart

Before you read on we like you to read this quickstart. The easiest way for many of the activities we do to interact with MongoDB is to use our cloudmesh functionality. This prelude section is not intended to describe all the details, but get you started quickly while leveraging cloudmesh

This is done via the cloudmesh cmd5 and the cloudmesh_community/cm code:

* https://cloudmesh-community.github.io/cm/

To install mongo on for example macOS you can use

```
$ cms admin mongo install
```

To start, stop and see the status of mongo you can use

```
$ cms admin mongo start
```

```
$ cms admin mongo stop
$ cms admin mongo status
```

To add an object to Mongo, you simply have to define a dict with predefined values for `kind` and `cloud`. In future such attributes can be passed to the function to determine the MongoDB collection.

```
from cloudmesh.mongo.DataBaseDecorator import DatabaseUpdate

@DatabaseUpdate
def test():
  data ={
    "kind": "test",
    "cloud": "testcloud",
    "value": "hello"
  }
  return data

result = test()
```

When you invoke the function it will automatically store the information into MongoDB. Naturally this requires that the `~/.cloudmesh/cloudmesh.yaml` file is properly configured.

3.7.2.2 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 [4]. At its core, MongoDB is an open source, cross-platform, document database mainly written in C++ language.

3.7.2.2.1 Installation

MongoDB can be installed on various Unix Platforms, including Linux, Ubuntu, Amazon Linux, etc [5]. 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.

3.7.2.2.1.1 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 [6].

```
$ 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 [6].

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 the next 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 [5].

```
$ 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 [5].

```
$ 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 [5].

3.7.2.2.2 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 [3].

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 [2].

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 [2].

3.7.2.2.2.1 Collection example

The following collection example for a person named *Albert* includes additional information such as age, status, and group [7].

```
{
 name: "Albert"
 age: "21"
 status: "Open"
 group: ["AI" , "Machine Learning"]
}
```

3.7.2.2.2.2 Document structure

```
{
    field1: value1,
    field2: value2,
    field3: value3,
    ...
    fieldN: valueN
}
```

3.7.2.2.2.3 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 } )
```

3.7.2.2.3 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 [3]. 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 [8]. MongoDB queries support regular expressions as well as range asks for specific fields that eliminate the need of returning entire documents [3]. 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[2].

3.7.2.2.3.1 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"* [9].

```
$ {
   $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: { $regex: /789$/ } } )
```

3.7.2.2.4 MongoDB Basic Functions

When it comes to the technical elements of MongoDB, it posses 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 [10].

3.7.2.2.4.1 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 [10].

```
$ 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"* [10].

```
$ mongoexport --db test --collection traffic --out traffic.json
```

3.7.2.2.5 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 ole 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 [11].

3.7.2.2.5.1 Collection based access control example

A user defined role can contain the following privileges [11].

```
$ privileges: [
   { resource: { db: "products", collection: "inventory" }, actions: [ "find", "update"] },
   { resource: { db: "products", collection: "orders" },  actions: [ "find" ] }
 ]
```

3.7.2.2.6 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 [12].

3.7.2.3 PyMongo

PyMongo is the official Python driver or distribution that allows work with a NoSQL type database called *MongoDB* [13]. The first version of the driver was developed in 2009 [14], 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 [15]. 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 [16]. For this reason, dictionaries are the recommended tools to be used in PyMongo when representing documents [17].

3.7.2.3.1 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 [18]:

```
$ 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 [18].

```
$ python -m pip install pymongo==3.5.1
```

A single line of code can be used to upgrade the driver as well [18].

```
$ 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 [18].

```
$ python -m easy_install pymongo
```

To do an upgrade of the driver using this tool, the following command is recommended [18]:

```
$ 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 [18].

To check if the installation was completed accurately, the following command is used in the Python console [19].

```
import pymongo
```

If the command returns zero exceptions within the Python shell, one can

consider for the PyMongo installation to have been completed successfully.

3.7.2.3.2 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 [15]. An optional dependency that requires some additional components to be installed is the GSSAPI authentication [15]. For the Unix based machines, it requires *pykerberos*, while for the Windows machines *WinKerberos* is needed to fullfill this requirement [15]. 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 *wincerstore* are necessary for connections with help of TLS/SSL and can also be simultaneously installed along with the driver installation [15].

3.7.2.3.3 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 [19].

```
$ mongod
```

3.7.2.3.4 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 sub-sequentially allows the MongoClient object to communicate with the database [19].

```
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 [19].

3.7.2.3.5 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 [19]. 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']
```

3.7.2.3.6 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 [20]. The example of this command is presented in the followng section:

```
import pymongo
client = pymongo.MongoClient('mongodb://localhost:27017/')
db = client['cloudmesh']
```

3.7.2.3.7 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 [19]. 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 [19]. An implementation of this method is given in the following example.

```
gregors_course = cloudmesh.find_one({'instructor':'Gregor von Laszewski'})
```

Similarly, to retieve 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 [19]. In order to return individual documents, iteration over the result must be completed [19].

3.7.2.3.8 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 [20]. This method takes in one parameter which specifies the number of documents to be returned [20]. 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)
```

3.7.2.3.9 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 [20]. 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 [20]. 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)
```

3.7.2.3.10 Counting Documents

Counting documents can be done with one simple operation called *count_documents()* instead of using a full query [21]. For example, we can count the documents in the *cloudmesh_commpunity* 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 contraints 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)
```

3.7.2.3.11 Indexing

Indexing is a very important part of querying. It can greately improve query performance but also add functionality and aide in storing documents [21].

> *"To create a unique index on a key that rejects documents whose value for that key already exists in the index" [21].*

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.

3.7.2.3.12 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 *pymongo.ascending* and *pymongo.descending* [22]. 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')
```

3.7.2.3.13 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"* [23].

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 24]. 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* [24]. 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": "$birthDay" }] }},
        "in": {
          "$switch": {
            "branches": [
              { "case": { "$lt": [ "$$age", 20 ] }, "then": 0 },
              { "case": { "$lt": [ "$$age", 30 ] }, "then": 20 },
              { "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 [24].

```
db.artwork.aggregate( [
  {
    $bucketAuto: {
```

```
        groupBy: "$price",
        buckets: 4
    }
  }
] )
```

The *collStats* stage returns statistics regarding a collection or view [24].

```
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 [24].

```
db.scores.aggregate(  [    {
   $match: {        score: {        $gt: 80    } }  },
{     $count: "passing_scores"  } ])
```

The *facet* stage helps process multiple aggregation pipelines in a single stage [24].

```
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 [24].

```
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 [24].

```
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 [24].

```
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 [24].

```
db.orders.aggregate( [ { $indexStats: { } } ] )
```

The *limit* stage is used for controlling the number of documents passed to the next stage in the pipeline [24].

```
db.article.aggregate(
  { $limit : 5 }
)
```

The *listLocalSessions* stage gives the session information currently connected to mongos or mongod instance [24].

```
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 [24].

```
 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 [24].

```
{
   $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 [24].

```
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 [24].

```
  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 [24].

```
  db.produce.aggregate( [
    {
      $replaceRoot: { newRoot: "$in_stock" }
    }
] )
```

The *sample* stage is used to sample out data by randomly selecting number of documents form input [24].

```
  db.users.aggregate(
    [ { $sample: { size: 3 } } ]
)
```

The *skip* stage skips specified initial number of documents and passes remaining documents to the pipeline [24].

```
db.article.aggregate(
    { $skip : 5 }
);
```

The *sort* stage is useful while reordering document stream by a specified sort key [24].

```
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 [24].

```
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 [24].

```
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 [24].

```
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"* [23].

The last step in the Map/Reduce process it to call the *map_reduce()* function and iterate over the results [23]. 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 [25]. 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 [25]. The PyMongo's API supports all features of the MongoDB's Map/Reduce engine [26]. Moreover, Map/Reduce has the ability to get more detailed results by passing *full_response=True* argument to the *map_reduce()* function [26].

3.7.2.3.14 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 [22]. 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 [22].

3.7.2.3.15 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 [27]. 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-prodected, one would not need to pass in the credentials nor to authenticate to the admin database [27]. 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')
```

3.7.2.3.16 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" [22].*

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" [22].*

According to [22], MongoDB is very applicable to modern applications, which makes PyMongo equally valuable [22].

3.7.2.4 MongoEngine

> *"MongoEngine is an Object-Document Mapper, written in Python for working with MongoDB" [28].*

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)" [19].*

The primary technique why one would use an ODM includes *data conversion* between computer systems that are not compatible with each other [29]. For the purpose of converting data to the appropriate form, a *virtual object database* must be created within the utilized programming language [29]. 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 [30]. It is also very beneficial to the overall workflow as it tracks changes made to the documents and aids in the document saving process [31].

3.7.2.4.1 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 [32]:

```
$ 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.

3.7.2.4.2 Connecting to a database using MongoEngine

Once installed, MongoEngine needs to be connected to an instance of the mongod, similarly to PyMongo [33]. 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 [33]. 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 [33].

```
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 [33].

3.7.2.4.3 Querying using MongoEngine

To query MongoDB using MongoEngine an *objects attribute* is used, which is, technically, a part of the document class [34]. This attribute is called the *QuerySetManager* which in return

> *"creates a new QuerySet object on access" [34].*

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 [34]. Let us 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 [34].

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=[[[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 MongoDBand which makes this pair even more attractive to application developers.

3.7.2.5 Flask-PyMongo

> *"Flask is a micro-web framework written in Python" [35].*

It was developed after Django, and it is very pythonic in nature which implies that it is explicitly the 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 [35]. 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 [36].

3.7.2.5.1 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)
```

3.7.2.5.2 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 confiuration variable"* [36].

3.7.2.5.3 Connection to multiple databases/servers

Multiple PyMongo instances can be used to connect to multiple databases or database servers. To achieve this, once would use a command similar to the following:

```python
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")
```

3.7.2.5.4 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:

```python
@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 [36].

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 [36].

3.7.2.5.5 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* [37].

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 [38].

3.7.2.5.6 Classes and Wrappers

Attributes such as *cx* and *db* in the PyMongo objects are the ones that help provide access to the MongoDB server [36]. To achieve this, one must pass the Flask app to the constructor or call *init_app()* [36].

> *"Flask-PyMongo wraps PyMongo's MongoClient, Database, and Collection classes, and overrides their attribute and item accessors" [36].*

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 [36].

```
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"* [39],

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 [39]. 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 [39].

3.7.3 Mongoengine ☁

3.7.3.1 Introduction

MongoEngine is a document mapper for working with mongoldb with python. To be able to use mongo engine MongodD should be already installed and running.

3.7.3.2 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 mongo engine import * connect ('student_db')
```

If mongodb is running on a port different from default port , port number and host need to be specified. If mongoldb needs authentication username and password need to be specified.

3.7.3.3 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.

```python
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 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).

```python
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 then storing them separately and then having them referenced.Similarly Comments can also be stored directly in a Text.

```python
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.

```python
for text in OnlyText.objects:
    print(text.title)
```

Searching texts with tags.

```python
for text in Text.objects(tags='mongodb'):
    print(text.title)
```

3.8 CALCULATION

3.8.1 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.

3.8.1.1 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:

```python
'''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
```

3.8.1.2 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)
```

3.8.1.3 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, 2]
print(map(str, nums))
```

```
['2', '1', '1', '1', '2', '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))
17
```

We can simplify this even more by using a lambda function:

```
print(reduce(lambda x, y: x + y, nums))
17
```

You can read more about [Python's lambda function in the docs](#).

With this in mind, we can reimplement the wordcount example as follows:

```python
'''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)
```

3.8.1.4 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)
```

3.8.1.5 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.

3.8.1.6 Excersises

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 goal from 1 to 2 processes? From 2 to 3? When do you stop seeing improvement? (this will depend on your machine architecture)*

3.8.1.7 References

- [Map, Filter and Reduce](#)
- [multiprocessing API](#)

3.8.2 NumPy ☁

NumPy is a popular library that is used by many other Python packages such as Pandas, SciPy, and scikit-learn. It provides a fast, simple-to-use way of interacting with numerical data organized in vectors and matrices. In this section, we will provide a short introduction to NumPy.

3.8.2.1 Installing NumPy

The most common way of installing NumPy, if it wasn't included with your Python installation, is to install it via pip:

```
$ pip install numpy
```

If NumPy has already been installed, you can update to the most recent version using:

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

You can verify that NumPy is installed by trying to use it in a Python program:

```
import numpy as np
```

Note that, by convention, we import NumPy using the alias 'np' - whenever you see 'np' sprinkled in example Python code, it's a good bet that it is using NumPy.

3.8.2.2 NumPy Basics

At its core, NumPy is a container for n-dimensional data. Typically, 1-dimensional data is called an array and 2-dimensional data is called a matrix. Beyond 2-dimenions would be considered a multidimensional array. Examples where you'll encounter these dimenions may include:

- 1 Dimensional: time series data such as audio, stock prices, or a single observation in a dataset.
- 2 Dimensional: connectivity data between network nodes, user-product recommendations, and database tables.
- 3+ Dimensional: network latency between nodes over time, video (RGB+time), and version controlled datasets.

All of these data can be placed into NumPy's array object, just with varying dimensions.

3.8.2.3 Data Types: The Basic Building Blocks

Before we delve into arrays and matrices, we will start off with the most basic element of those: a single value. NumPy can represent data utilizing many different standard datatypes such as uint8 (an 8-bit **u**signed **int**eger), float64 (a 64-bit float), or str (a string). An exhaustive listing can be found at:

- https://docs.scipy.org/doc/numpy-1.15.0/user/basics.types.html

Before moving on, it is important to know about the tradeoff made when using different datatypes. For example, a uint8 can only contain values between 0 and 255. This, however, contrasts with float64 which can express any value from +/-

1.80e+308. So why wouldn't we just always use float64s? Though they allow us to be more expressive in terms of numbers, they also consume more memory. If we were working with a 12 megapixel image, for example, storing that image using uint8 values would require 3000 * 4000 * 8 = 96 million bits, or 91.55 MB of memory. If we were to store the same image utilizing float64, our image would consume 8 times as much memory: 768 million bits or 732.42 MB. It's important use the right datatype for the job to avoid consuming unneccessary resources or slowing down processing.

Finally, while NumPy will conveniently convert between datatypes, one must be aware of overflows when using smaller datatypes. For example:

```
a = np.array([6], dtype=np.uint8)
print(a)
>>>[6]
a = a + np.array([7], dtype=np.uint8)
print(a)
>>>[13]
a = a + np.array([245], dtype=np.uint8)
print(a)
>>>[2]
```

In this example, it makes sense that 6+7=13. But how does 13+245=2? Put simply, the object type (uint8) simply ran out of space to store the value and wrapped back around to the beginning. An 8-bit number is only capable of storing 2^8, or 256, unique values. An operation that results in a value above that range will 'overflow' and cause the value to wrap back around to zero. Likewise, anything below that range will 'underflow' and wrap back around to the end. In our example, 13+245 became 258, which was too large to store in 8 bits and wrapped back around to 0 and ended up at 2.

NumPy will, generally, try to avoid this situation by dynamically retyping to whatever datatype will support the result:

```
a = a + 260
print(test)
>>>[262]
```

Here, our addition caused our array, 'a', to be upscaled to use uint16 instead of uint8. Finally, NumPy offers convenience functions akin to Python's range() function to create arrays of sequential numbers:

```
X = np.arange(0.2,1,.1)
print(X)
>>>array([0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9], dtype=float32)
```

We can use this function to also generate parameters spaces that can 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))
```

3.8.2.4 Arrays: Stringing Things Together

With our knowledge of datatypes in hand, we can begin to explore arrays. Simply put, arrays can be thought of as a sequence of values (not neccesarily numbers). Arrays are 1 dimensional and can be created and accessed simply:

```
a = np.array([1, 2, 3])
print(type(a))
>>><class 'numpy.ndarray'>
print(a)
>>>[1 2 3]
print(a.shape)
>>>(3,)
a[0]
>>>1
```

Arrays (and, later, matrices) are zero-indexed. This makes it convenient when, for example, using Python's range() function to iterate through an array:

```
for i in range(3):
    print(a[i])
>>>1
>>>2
>>>3
```

Arrays are, also, mutable and can be changed easily:

```
a[0] = 42
print(a)
>>>array([42, 2, 3])
```

NumPy also includes incredibly powerful broadcasting features. This makes it very simple to perform mathematical operations on arrays that also makes intuitive sense:

```
a * 3
>>>array([3, 6, 9])
a**2
>>>array([1, 4, 9], dtype=int32)
```

Arrays can also interact with other arrays:

```
b = np.array([2, 3, 4])
print(a * b)
>>>array([ 2,  6, 12])
```

In this example, the result of multiplying together two arrays is to take the element-wise product while multiplying by a constant will multiply each element in the array by that constant. NumPy supports all of the basic mathematical operations: addition, subtraction, multiplication, division, and powers. It also includes an extensive suite of mathematical functions, such as log() and max(), which are covered later.

3.8.2.5 Matrices: An Array of Arrays

Matrices can be thought of as an extension of arrays - rather than having one dimension, matrices have 2 (or more). Much like arrays, matrices can be created easily within NumPy:

```
m = np.array([[1, 2], [3, 4]])
print(m)
>>>[[1 2]
>>> [3 4]]
```

Accessing individual elements is similar to how we did it for arrays. We simply need to pass in a number of arguments equal to the number of dimensions:

```
m[1][0]
>>>3
```

In this example, our first index selected the row and the second selected the column - giving us our result of 3. Matrices can be extending out to any number of dimensions by simply using more indices to access specific elements (though use-cases beyond 4 may be somewhat rare).

Matrices support all of the normal mathematial functions such as +, -, *, and /. A special note: the * operator will result in an element-wise multiplication. Using @ or np.matmul() for matrix multiplication:

```
print(m-m)
print(m*m)
print(m/m)
```

More complex mathematical functions can typically be found within the NumPy library itself:

```
print(np.sin(x))
print(np.sum(x))
```

A   full   listing   can   be   found   at:

https://docs.scipy.org/doc/numpy/reference/routines.math.html

3.8.2.6 Slicing Arrays and Matrices

As one can imagine, accessing elements one-at-a-time is both slow and can potentially require many lines of code to iterate over every dimension in the matrix. Thankfully, NumPy incorporate a very powerful slicing engine that allows us to access ranges of elements easily:

```
m[1, :]
>>>array([3, 4])
```

The ':' value tells NumPy to select all elements in the given dimension. Here, we've requested all elements in the first row. We can also use indexing to request elements within a given range:

```
a = np.arange(0, 10, 1)
print(a)
>>>[0 1 2 3 4 5 6 7 8 9]
a[4:8]
>>>array([4, 5, 6, 7])
```

Here, we asked NumPy to give us elements 4 through 7 (ranges in Python are inclusive at the start and non-inclusive at the end). We can even go backwards:

```
a[-5:]
>>>array([5, 6, 7, 8, 9])
```

In the previous example, the negative value is asking NumPy to return the last 5 elements of the array. Had the argument been ':-5', NumPy would've returned everything BUT the last five elements:

```
a[:-5]
>>>array([0, 1, 2, 3, 4])
```

Becoming more familiar with NumPy's accessor conventions will allow you write more efficient, clearer code as it is easier to read a simple one-line accessor than it is a multi-line, nested loop when extracting values from an array or matrix.

3.8.2.7 Useful Functions

The NumPy library provides several convenient mathematical functions that users can use. These functions provide several advantages to code written by

users:

- They are open source typically have multiple contributors checking for errors.
- Many of them utilize a C interface and will run much faster than native Python code.
- They're written to very flexible.

NumPy arrays and matrices contain many useful aggregating functions such as max(), min(), mean(), etc These functions are usually able to run an order of magnitude faster than looping through the object, so it's important to understand what functions are available to avoid 'reinventing the wheel.' In addition, many of the functions are able to sum or average across axes, which make them extremely useful if your data has inherent grouping. To return to a previous example:

```
m = np.array([[1, 2], [3, 4]])
print(m)
>>>[[1 2]
>>> [3 4]]
m.sum()
>>>10
m.sum(axis=1)
>>>[3, 7]
m.sum(axis=0)
>>>[4, 6]
```

In this example, we created a 2x2 matrix containing the numbers 1 through 4. The sum of the matrix returned the element-wise addition of the entire matrix. Summing across axis 0 (rows) returned a new array with the element-wise addition across each row. Likewise, summing across axis 1 (columns) returned the columnar summation.

3.8.2.8 Linear Algebra

Perhaps one of the most important uses for NumPy is its robust support for Linear Algebra functions. Like the aggregation functions described in the previous section, these functions are optimized to be much faster than user implementations and can utilize processesor level features to provide very quick computations. These functions can be accessed very easily from the NumPy package:

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
print(np.matmul(a, b))
```

```
>>>[[19 22]
   [43 50]]
```

Included in within np.linalg are functions for calculating the Eigendecomposition of square matrices and symmetric matrices. Finally, to give a quick example of how easy it is to implement algorithms in NumPy, we can easily use it to calculate the cost and gradient when using simple Mean-Squared-Error (MSE):

```
cost = np.power(Y - np.matmul(X, weights)), 2).mean(axis=1)
gradient = np.matmul(X.T, np.matmul(X, weights) - y)
```

Finally, more advanced functions are easily available to users via the linalg library of NumPy as:

```
from numpy import linalg

A = np.diag((1,2,3))

w,v = linalg.eig(A)

print ('w =', w)
print ('v =', v)
```

3.8.2.9 NumPy Resources

- https://docs.scipy.org/doc/numpy
- http://cs231n.github.io/python-numpy-tutorial/#numpy
- https://docs.scipy.org/doc/numpy-1.15.1/reference/routines.linalg.html
- https://en.wikipedia.org/wiki/Mean_squared_error

3.8.3 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 bundels 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

3.8.3.1 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.

```python
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.

```python
s = sp.randn(100)
```

How many elements are in the set?

```python
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 %0.16f' %
       (abs(sp.std(s)**2 - sp.var(s))))
```

How does this look as a histogram? See Figure 18, Figure 19, Figure 20

```
plt.hist(s) # yes, one line of code for a histogram
plt.show()
```
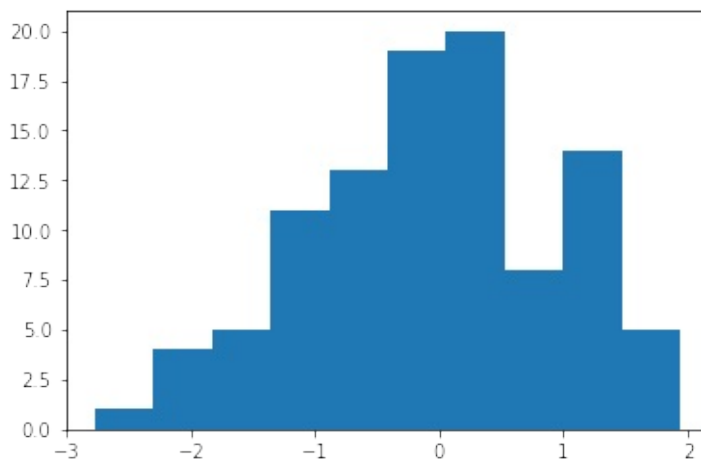
Figure 18: Histogram 1

Let us 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()
```
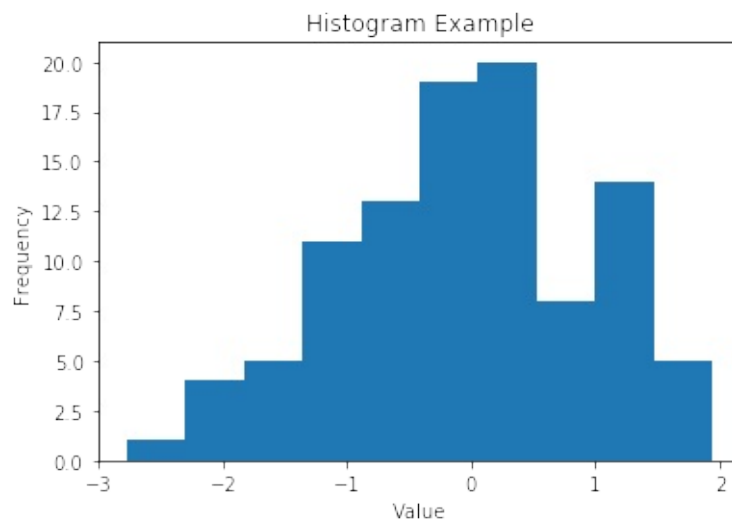

Figure 19: Histogram 2

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()
```
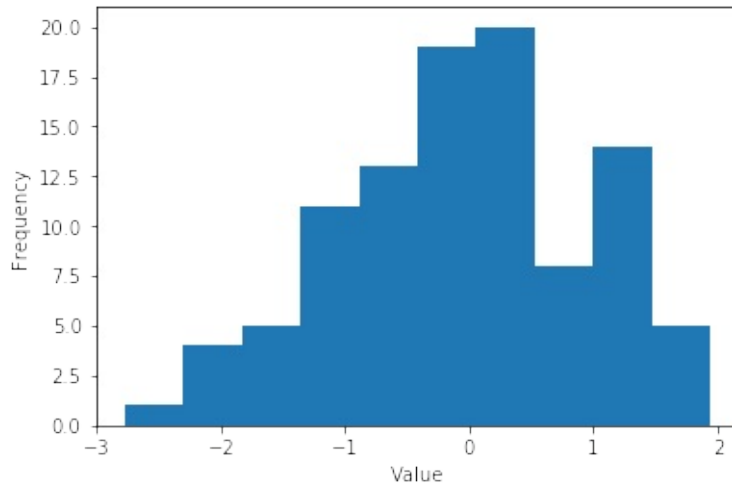
Figure 20: Histogram 3

Let us 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()
```
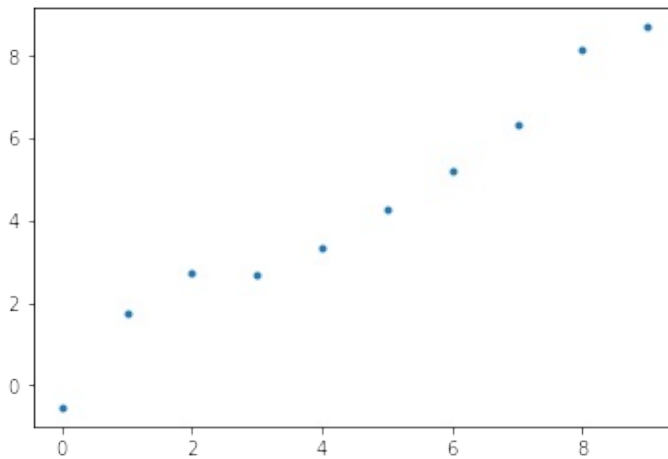

Figure 21: Result 1

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(['obsered 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 22](#)

```
plt.savefig("regression.pdf", bbox_inches='tight')

plt.savefig('regression.png')

plt.show()
```
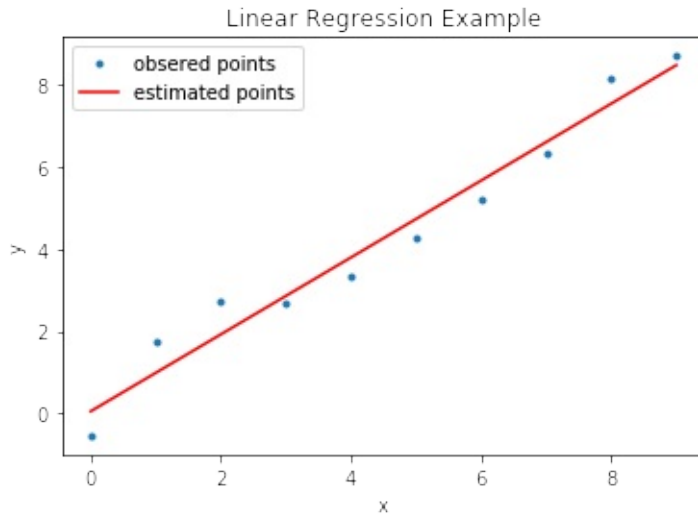
Figure 22: Result 2

3.8.3.2 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

- [⊗] "Getting Started guide" https://www.scipy.org/getting-started.html
- [⊗] Prasanth. "Simple statistics with SciPy." Comfort at 1 AU. February 28, 2011. https://oneau.wordpress.com/2011/02/28/simple-statistics-with-scipy/.
- [⊗] SciPy Cookbook. Lasted updated: 2015. http://scipy-cookbook.readthedocs.io/.

⊗ create bibtex entries

3.8.4 Scikit-learn ☁

 Learning Objectives

- Exploratory data analysis
- Pipeline to prepare data
- Full learning pipeline
- Fine tune the model
- Significance tests

3.8.4.1 Introduction to Scikit-learn

Scikit learn is a Machine Learning specific library used in Python. Library can be used for data mining and analysis. It is built on top of NumPy, matplotlib and SciPy. Scikit Learn features Dimensionality reduction, clustering, regression and classification algorithms. It also features model selection using grid search, cross validation and metrics.

Scikit learn also enables users to preprocess the data which can then be used for machine learning using modules like preprocessing and feature extraction.

In this section we demonstrate how simple it is to use k-means in scikit learn.

3.8.4.2 Installation

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
```

3.8.4.3 Supervised Learning

Supervised Learning is used in machine learning when we already know a set of output predictions based on input characteristics and based on that we need to predict the target for a new input. Training data is used to train the model which then can be used to predict the output from a bounded set.

Problems can be of two types

1. Classification : Training data belongs to three or four classes/categories and based on the label we want to predict the class/category for the unlabeled data.
2. Regression : Training data consists of vectors without any corresponding target values. Clustering can be used for these type of datasets to determine discover groups of similar examples. Another way is density estimation which determine the distribution of data within the input space. Histogram is the most basic form.

3.8.4.4 Unsupervised Learning

Unsupervised Learning is used in machine learning when we have the training set available but without any corresponding target. The outcome of the problem is to discover groups within the provided input. It can be done in many ways.

Few of them are listed here

1. Clustering : Discover groups of similar characteristics.
2. Density Estimation : Finding the distribution of data within the provided input or changing the data from a high dimensional space to two or three dimension.

3.8.4.5 Building a end to end pipeline for Supervised machine learning using Scikit-learn

A data pipeline is a set of processing components that are sequenced to produce meaningful data. Pipelines are commonly used in Machine learning, since there is lot of data transformation and manipulation that needs to be applied to make data useful for machine learning. All components are sequenced in a way that the output of one component becomes input for the next and each of the component is self contained. Components interact with each other using data.

Even if a component breaks, the downstream component can run normally using the last output. Sklearn provide the ability to build pipelines that can be transformed and modeled for machine learning.

3.8.4.6 Steps for developing a machine learning model

1. Explore the domain space
2. Extract the problem definition
3. Get the data that can be used to make the system learn to solve the problem definition.
4. Discover and Visualize the data to gain insights
5. Feature engineering and prepare the data
6. Fine tune your model
7. Evaluate your solution using metrics
8. Once proven launch and maintain the model.

3.8.4.7 Exploratory Data Analysis

Example project = Fraud detection system

First step is to load the data into a dataframe in order for a proper analysis to be done on the attributes.

```
data = pd.read_csv('dataset/data_file.csv')
data.head()
```

Perform the basic analysis on the data shape and null value information.

```
print(data.shape)
print(data.info())
data.isnull().values.any()
```

Here is the example of few of the visual data analysis methods.

3.8.4.7.1 Bar plot

A bar chart or graph is a graph with rectangular bars or bins that are used to plot categorical values. Each bar in the graph represents a categorical variable and the height of the bar is proportional to the value represented by it.

Bar graphs are used:

To make comparisons between variables To visualize any trend in the data, i.e., they show the dependence of one variable on another Estimate values of a variable

```
plt.ylabel('Transactions')
plt.xlabel('Type')
data.type.value_counts().plot.bar()
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1a7891fb70>
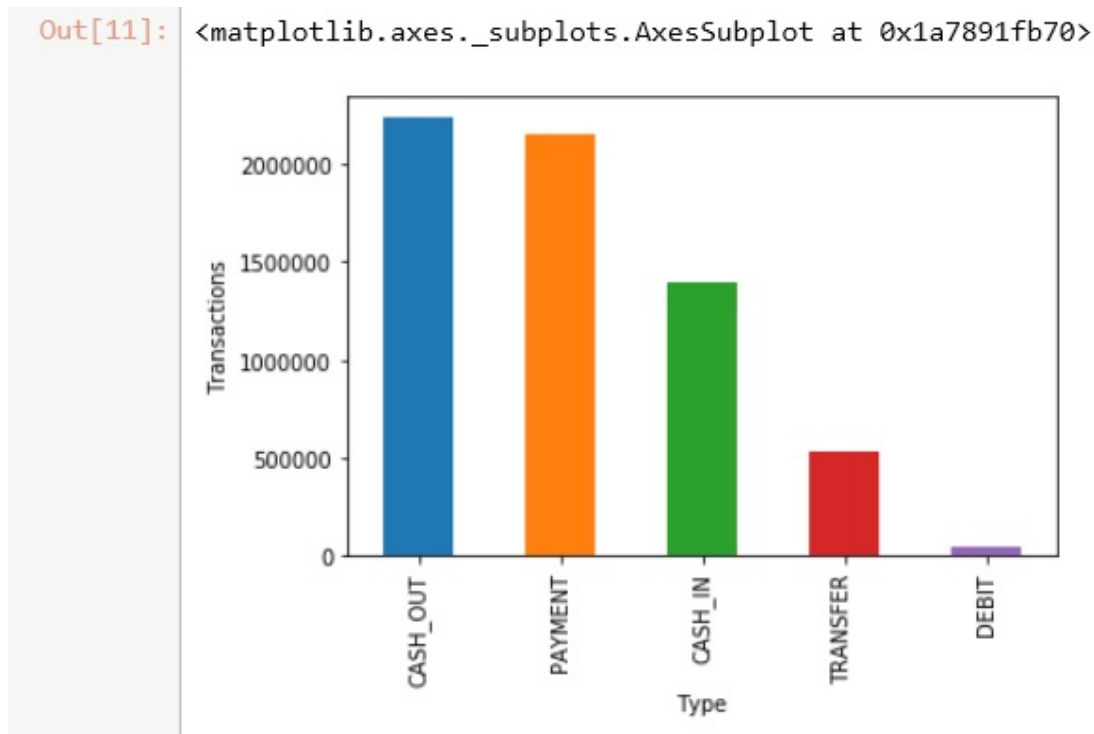
Figure 23: Example of scikit-learn barplots

3.8.4.7.2 Correlation between attributes

Attributes in a dataset can be related based on differnt aspects.

Examples include attributes dependent on another or could be loosely or tightly coupled. Also example includes two variables can be associated with a third one.

In order to understand the relationship between attributes, correlation represents the best visual way to get an insight. Positive correlation meaning both attributes moving into the same direction. Negative correlation refers to opposte directions. One attributes values increase results in value decrease for other. Zero correlation is when the attributes are unrelated.

```
# compute the correlation matrix
corr = data.corr()

# generate a mask for the lower triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# set up the matplotlib figure
f, ax = plt.subplots(figsize=(18, 18))
```

```
# generate a custom diverging color map
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3,
            square=True,
            linewidths=.5, cbar_kws={"shrink": .5}, ax=ax);
```
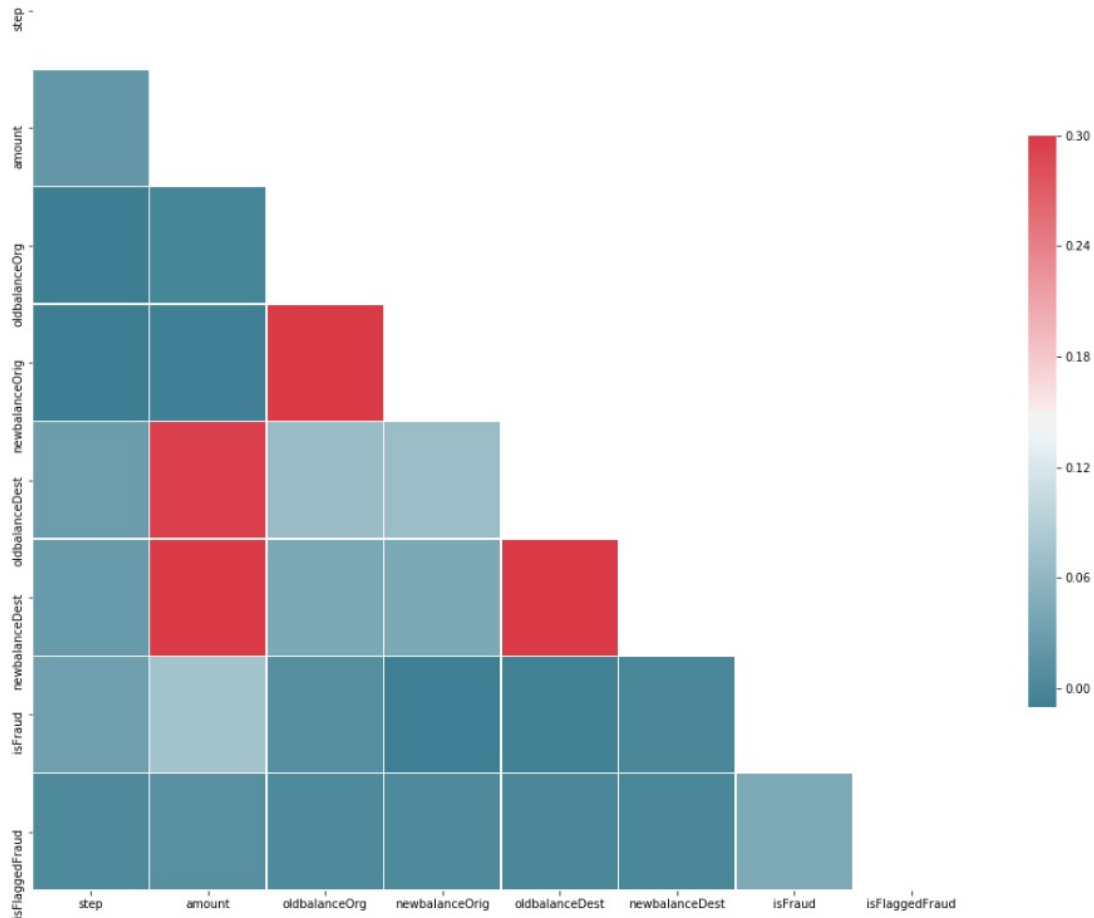

Figure 24: scikit-learn correlation array

3.8.4.7.3 Histogram Analysis of dataset attributes

A histogram consists of a set of counts that represent the number of times some event occurred.

```
%matplotlib inline
data.hist(bins=30, figsize=(20,15))
plt.show()
```
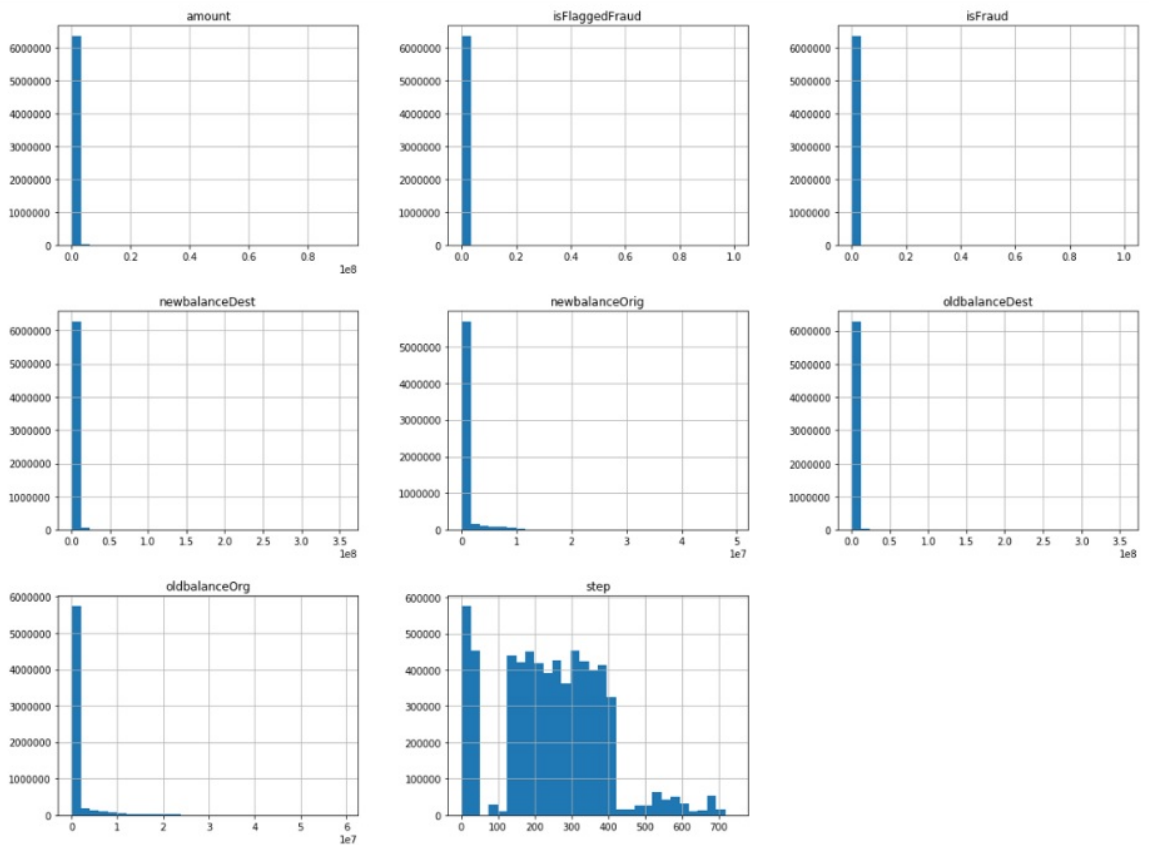
Figure 25: scikit-learn

3.8.4.7.4 Box plot Analysis

Box plot analysis is useful in detecting whether a distribution is skewed and detect outliers in the data.

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
tmp = data.loc[(data.type == 'TRANSFER'), :]

a = sns.boxplot(x = 'isFlaggedFraud', y = 'amount', data = tmp, ax=axs[0][0])
axs[0][0].set_yscale('log')
b = sns.boxplot(x = 'isFlaggedFraud', y = 'oldbalanceDest', data = tmp, ax=axs[0][1])
axs[0][1].set(ylim=(0, 0.5e8))
c = sns.boxplot(x = 'isFlaggedFraud', y = 'oldbalanceOrg', data=tmp, ax=axs[1][0])
axs[1][0].set(ylim=(0, 3e7))
d = sns.regplot(x = 'oldbalanceOrg', y = 'amount', data=tmp.loc[(tmp.isFlaggedFraud ==1), :], ax=axs[1][1])
plt.show()
```
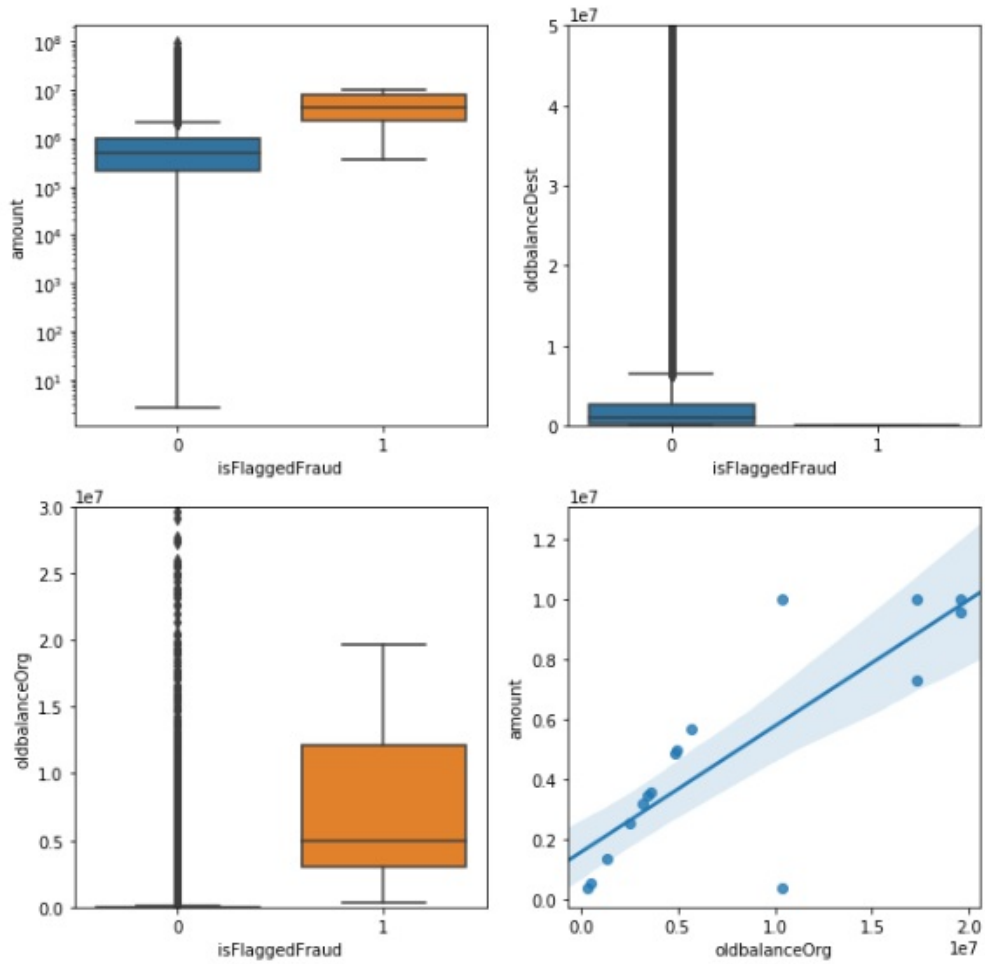
Figure 26: scikit-learn

3.8.4.7.5 Scatter plot Analysis

The scatter plot displays values of two numerical variables as Cartesian coordinates.

```
plt.figure(figsize=(12,8))
sns.pairplot(data[['amount', 'oldbalanceOrg', 'oldbalanceDest', 'isFraud']], hue='isFraud')
```
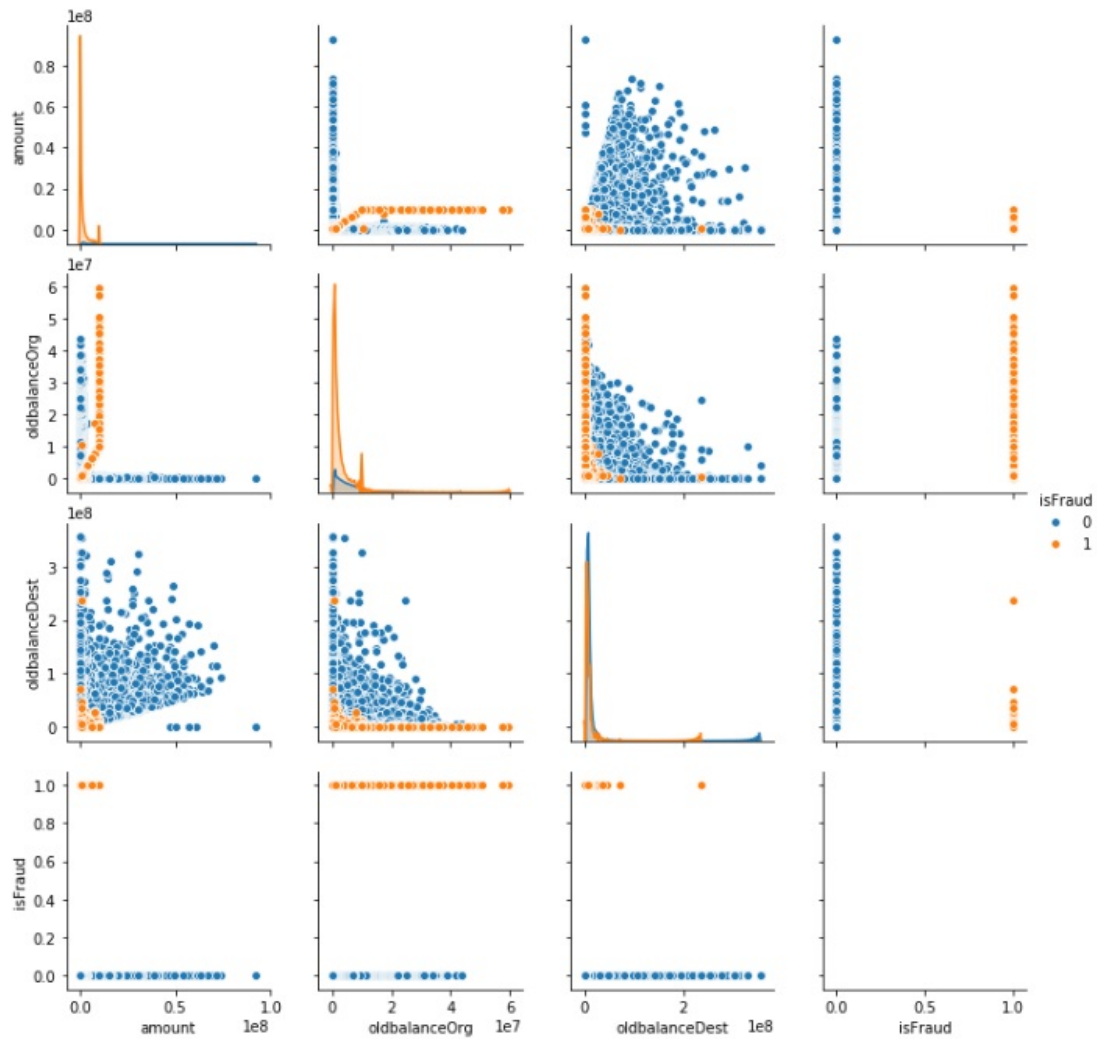
Figure 27: scikit-learn scatter plots

3.8.4.8 Data Cleansing - Removing Outliers

If the transaction amount is lower than 5 percent of the all the transactions AND does not exceed USD 3000, we will exclude it from our analysis to reduce Type 1 costs If the transaction amount is higher than 95 percent of all the transactions AND exceeds USD 500000, we will exclude it from our analysis, and use a blanket review process for such transactions (similar to isFlaggedFraud column in original dataset) to reduce Type 2 costs

```
low_exclude = np.round(np.minimum(fin_samp_data.amount.quantile(0.05), 3000), 2)
high_exclude = np.round(np.maximum(fin_samp_data.amount.quantile(0.95), 500000), 2)

###Updating Data to exclude records prone to Type 1 and Type 2 costs
low_data = fin_samp_data[fin_samp_data.amount > low_exclude]
```

```
data = low_data[low_data.amount < high_exclude]
```

3.8.4.9 Pipeline Creation

Machine learning pipeline is used to help automate machine learning workflows. They operate by enabling a sequence of data to be transformed and correlated together in a model that can be tested and evaluated to achieve an outcome, whether positive or negative.

3.8.4.9.1 Defining DataFrameSelector to separate Numerical and Categorical attributes

Sample function to seperate out Numerical and categorical attributes.

```
from sklearn.base import BaseEstimator, TransformerMixin

# Create a class to select numerical or categorical columns
# since Scikit-Learn doesn't handle DataFrames yet
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

3.8.4.9.2 Feature Creation / Additional Feature Engineering

During EDA we identified that there are transactions where the balances do not tally after the transaction is completed.We believe this could potentially be cases where fraud is occurring. To account for this error in the transactions, we define two new features"errorBalanceOrig" and "errorBalanceDest", calculated by adjusting the amount with the before and after balances for the Originator and Destination accounts.

Below, we create a function that allows us to create these features in a pipeline.

```
from sklearn.base import BaseEstimator, TransformerMixin

# column index
amount_ix, oldbalanceOrg_ix, newbalanceOrig_ix, oldbalanceDest_ix, newbalanceDest_ix = 0, 1, 2, 3, 4

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self): # no *args or **kargs
        pass
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X, y=None):
        errorBalanceOrig = X[:,newbalanceOrig_ix] + X[:,amount_ix] - X[:,oldbalanceOrg_ix]
        errorBalanceDest = X[:,oldbalanceDest_ix] + X[:,amount_ix]- X[:,newbalanceDest_ix]

        return np.c_[X, errorBalanceOrig, errorBalanceDest]
```

3.8.4.10 Creating Training and Testing datasets

Training set includes the set of input examples that the model will be fit into or trained on by adjusting the parameters. Testing dataset is critical to test the generalizability of the model . By using this set, we can get the working accuracy of our model.

Testing set should not be exposed to model unless model training has not been completed. This way the results from testing will be more reliable.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30, random_state=42, stratify=y)
```

3.8.4.11 Creating pipeline for numerical and categorical attributes

Identifying columns with Numerical and Categorical characteristics.

```
X_train_num = X_train[["amount","oldbalanceOrg", "newbalanceOrig", "oldbalanceDest", "newbalanceDest"]]
X_train_cat = X_train[["type"]]
X_model_col = ["amount","oldbalanceOrg", "newbalanceOrig", "oldbalanceDest", "newbalanceDest","type"]
```

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Imputer

num_attribs = list(X_train_num)
cat_attribs = list(X_train_cat)

num_pipeline = Pipeline([
        ('selector', DataFrameSelector(num_attribs)),
        ('attribs_adder', CombinedAttributesAdder()),
        ('std_scaler', StandardScaler())
    ])

cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        ('cat_encoder', CategoricalEncoder(encoding="onehot-dense"))
    ])
```

3.8.4.12 Selecting the algorithm to be applied

Algorithim selection primarily depends on the objective you are trying to solve and what kind of dataset is available. There are differnt type of algorithms which can be applied and we will look into few of them here.

3.8.4.12.1 Linear Regression

This algorithm can be applied when you want to compute some continuous value. To predict some future value of a process which is currently running, you

can go with regression algorithm.

Examples where linear regression can used are :

1. Predict the time taken to go from one place to another
2. Predict the sales for a future month
3. Predict sales data and improve yearly projections.

```python
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
import time
scl= StandardScaler()
X_train_std = scl.fit_transform(X_train)
X_test_std = scl.transform(X_test)
start = time.time()
lin_reg = LinearRegression()
lin_reg.fit(X_train_std, y_train) #SKLearn's linear regression
y_train_pred = lin_reg.predict(X_train_std)
train_time = time.time()-start
```

3.8.4.12.2 Logistic Regression

This algorithm can be used to perform binary classification. It can be used if you want a probabilistic framework. Also in case you expect to receive more training data in the future that you want to be able to quickly incorporate into your model.

1. Customer churn prediction.
2. Credit Scoring & Fraud Detection which is our example problem which we are trying to solve in this chapter.
3. Calculating the effectiveness of marketing campaigns.

```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X_train, _, y_train, _ = train_test_split(X_train, y_train, stratify=y_train, train_size=subsample_rate, random_state=42)
X_test, _, y_test, _ = train_test_split(X_test, y_test, stratify=y_test, train_size=subsample_rate, random_state=42)

model_lr_sklearn = LogisticRegression(multi_class="multinomial", C=1e6, solver="sag", max_iter=15)
model_lr_sklearn.fit(X_train, y_train)

y_pred_test = model_lr_sklearn.predict(X_test)
acc = accuracy_score(y_test, y_pred_test)
results.loc[len(results)] = ["LR Sklearn", np.round(acc, 3)]
results
```

3.8.4.12.3 Decision trees

Decision trees handle feature interactions and they're non-parametric. Doesnt support online learning and the entire tree needs to be rebuild when new traning

dataset comes in. Memory consumption is very high.

Can be used for the following cases

1. Investment decisions
2. Customer churn
3. Banks loan defaulters
4. Build vs Buy decisions
5. Sales lead qualifications

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
start = time.time()
dt.fit(X_train_std, y_train)
y_train_pred = dt.predict(X_train_std)
train_time = time.time() - start

start = time.time()
y_test_pred = dt.predict(X_test_std)
test_time = time.time() - start
```

3.8.4.12.4 K Means

This algorithm is used when we are not aware of the labels and one needs to be created based on the features of objects. Example will be to divide a group of people into differnt subgroups based on common theme or attribute.

The main disadvantage of K-mean is that you need to know exactly the number of clusters or groups which is required. It takes a lot of iteration to come up with the best K.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, PredefinedSplit
from sklearn.metrics import accuracy_score

X_train, _, y_train, _ = train_test_split(X_train, y_train, stratify=y_train, train_size=subsample_rate, random_state=42)
X_test, _, y_test, _ = train_test_split(X_test, y_test, stratify=y_test, train_size=subsample_rate, random_state=42)

model_knn_sklearn = KNeighborsClassifier(n_jobs=-1)
model_knn_sklearn.fit(X_train, y_train)

y_pred_test = model_knn_sklearn.predict(X_test)
acc = accuracy_score(y_test, y_pred_test)

results.loc[len(results)] = ["KNN Arbitary Sklearn", np.round(acc, 3)]
results
```

3.8.4.12.5 Support Vector Machines

SVM is a supervised ML technique and used for pattern recognition and

classification problems when your data has exactly two classes. Its popular in text classification problems.

Few cases where SVM can be used is

1. Detecting persons with common diseases.
2. Hand-written character recognition
3. Text categorization
4. Stock market price prediction

3.8.4.12.6 Naive Bayes

Naive Bayes is used for large datasets.This algoritm works well even when we have a limited CPU and memory available. This works by calculating bunch of counts. It requires less training data. The algorthim cant learn interation between features.

Naive Bayes can be used in real-world applications such as:

1. Sentiment analysis and text classification
2. Recommendation systems like Netflix, Amazon
3. To mark an email as spam or not spam
4. Face recognition

3.8.4.12.7 Random Forest

Ranmdon forest is similar to Decision tree. Can be used for both regression and classification problems with large data sets.

Few case where it can be applied.

1. Predict patients for high risks.
2. Predict parts failures in manufacturing.
3. Predict loan defaulters.

```
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor(n_estimators = 400, criterion='mse',random_state=1, n_jobs=-1)
start = time.time()
forest.fit(X_train_std, y_train)
y_train_pred = forest.predict(X_train_std)
train_time = time.time() - start

start = time.time()
```

```
y_test_pred = forest.predict(X_test_std)
test_time = time.time() - start
```

3.8.4.12.8 Neural networks

Neural network works based on weights of connections between neurons. Weights are trained and based on that the neural network can be utilized to predict the class or a quantity. They are resource and memory intensive.

Few cases where it can be applied.

1. Applied to unsupervised learning tasks, such as feature extraction.
2. Extracts features from raw images or speech with much less human intervention

3.8.4.12.9 Deep Learning using Keras

Keras is most powerful and easy-to-use Python libraries for developing and evaluating deep learning models. It has the efficient numerical computation libraries Theano and TensorFlow.

3.8.4.12.10 XGBoost

XGBoost stands for eXtreme Gradient Boosting. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. It is engineered for efficiency of compute time and memory resources.

3.8.4.13 Scikit Cheat Sheet

Scikit learning has put a very indepth and well explained flow chart to help you choose the right algorithm that I find very handy.
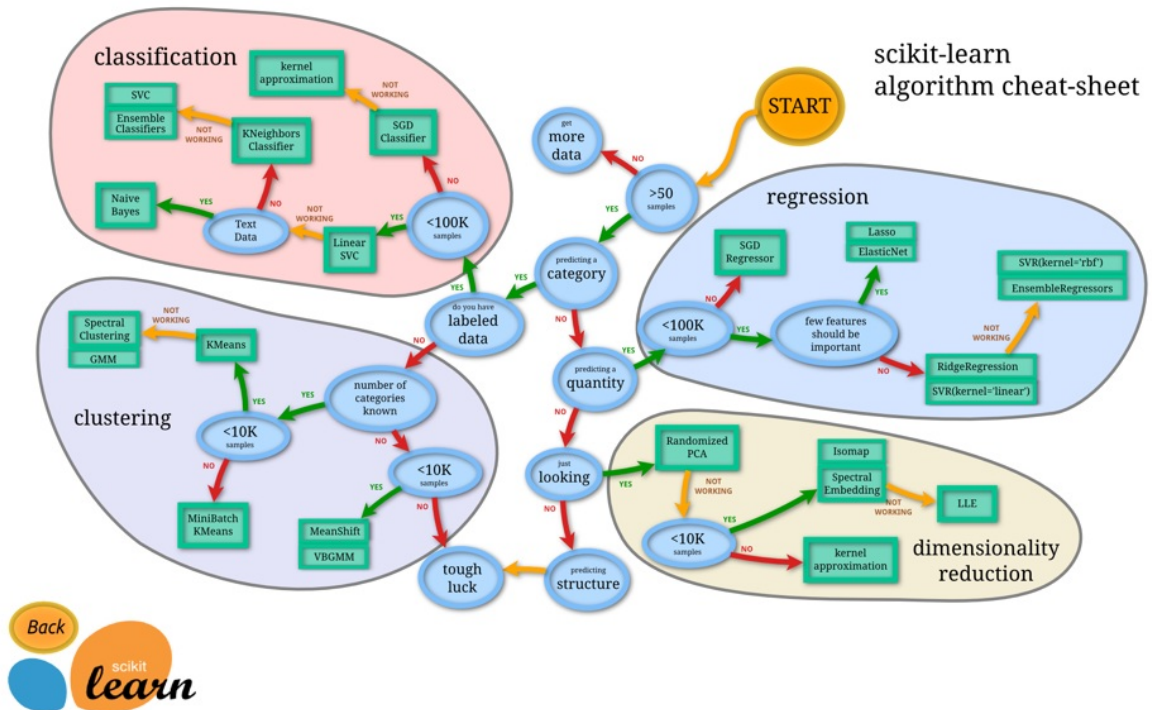
Figure 28: scikit-learn

3.8.4.14 Parameter Optimization

Machine learning models are parameterized so that their behavior can be tuned for a given problem. These models can have many parameters and finding the best combination of parameters can be treated as a search problem.

A parameter is a configurationthat is part of the model and values can be derived from the given data.

1. Required by the model when making predictions.
2. Values define the skill of the model on your problem.
3. Estimated or learned from data.
4. Often not set manually by the practitioner.
5. Often saved as part of the learned model.

3.8.4.14.1 Hyperparameter optimization/tuning algorithms

Grid search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in

a grid.

Random search provide a statistical distribution for each hyperparameter from which values may be randomly sampled.

3.8.4.15 Experiments with Keras (deep learning), XGBoost, and SVM (SVC) compared to Logistic Regression(Baseline)

3.8.4.15.1 Creating a parameter grid

```python
grid_param = [
            [{   #LogisticRegression
                'model__penalty':['l1','l2'],
                'model__C': [0.01, 1.0, 100]
            }],

            [{#keras
                'model__optimizer': optimizer,
                'model__loss': loss
            }],

            [{  #SVM
                'model__C' :[0.01, 1.0, 100],
                'model__gamma': [0.5, 1],
                'model__max_iter':[-1]
            }],

            [{   #XGBClassifier
                'model__min_child_weight': [1, 3, 5],
                'model__gamma': [0.5],
                'model__subsample': [0.6, 0.8],
                'model__colsample_bytree': [0.6],
                'model__max_depth': [3]

            }]
        ]
```

3.8.4.15.2 Implementing Grid search with models and also creating metrics from each of the model.

```python
Pipeline(memory=None,
     steps=[('preparation', FeatureUnion(n_jobs=None,
       transformer_list=[('num_pipeline', Pipeline(memory=None,
     steps=[('selector', DataFrameSelector(attribute_names=['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest'
          tol=0.0001, verbose=0, warm_start=False))])
```

```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from xgboost.sklearn import XGBClassifier
from sklearn.svm import SVC

test_scores = []
#Machine Learning Algorithm (MLA) Selection and Initialization
MLA = [
        linear_model.LogisticRegression(),
        keras_model,
        SVC(),
        XGBClassifier()

    ]
```

```
#create table to compare MLA metrics
MLA_columns = ['Name', 'Score', 'Accuracy_Score','ROC_AUC_score','final_rmse','Classification_error','Recall_Score','Precision_Score'
MLA_compare = pd.DataFrame(columns = MLA_columns)
Model_Scores = pd.DataFrame(columns = ['Name','Score'])


row_index = 0
for alg in MLA:

    #set name and parameters
    MLA_name = alg.__class__.__name__
    MLA_compare.loc[row_index, 'Name'] = MLA_name
    #MLA_compare.loc[row_index, 'Parameters'] = str(alg.get_params())


    full_pipeline_with_predictor = Pipeline([
        ("preparation", full_pipeline),  # combination of numerical and categorical pipelines
        ("model", alg)
    ])

    grid_search = GridSearchCV(full_pipeline_with_predictor, grid_param[row_index], cv=4, verbose=2, scoring='f1', return_train_score

    grid_search.fit(X_train[X_model_col], y_train)
    y_pred = grid_search.predict(X_test)

    MLA_compare.loc[row_index, 'Accuracy_Score'] = np.round(accuracy_score(y_pred, y_test), 3)
    MLA_compare.loc[row_index, 'ROC_AUC_score'] = np.round(metrics.roc_auc_score(y_test, y_pred),3)
    MLA_compare.loc[row_index,'Score'] = np.round(grid_search.score(X_test, y_test),3)

    negative_mse = grid_search.best_score_
    scores = np.sqrt(-negative_mse)
    final_mse = mean_squared_error(y_test, y_pred)
    final_rmse = np.sqrt(final_mse)
    MLA_compare.loc[row_index, 'final_rmse'] = final_rmse

    confusion_matrix_var = confusion_matrix(y_test, y_pred)
    TP = confusion_matrix_var[1, 1]
    TN = confusion_matrix_var[0, 0]
    FP = confusion_matrix_var[0, 1]
    FN = confusion_matrix_var[1, 0]
    MLA_compare.loc[row_index,'Classification_error'] = np.round(((FP + FN) / float(TP + TN + FP + FN)), 5)
    MLA_compare.loc[row_index,'Recall_Score'] = np.round(metrics.recall_score(y_test, y_pred), 5)
    MLA_compare.loc[row_index,'Precision_Score'] = np.round(metrics.precision_score(y_test, y_pred), 5)
    MLA_compare.loc[row_index,'F1_Score'] = np.round(f1_score(y_test,y_pred), 5)


    MLA_compare.loc[row_index, 'mean_test_score'] = grid_search.cv_results_['mean_test_score'].mean()
    MLA_compare.loc[row_index, 'mean_fit_time'] = grid_search.cv_results_['mean_fit_time'].mean()

    Model_Scores.loc[row_index,'MLA Name'] = MLA_name
    Model_Scores.loc[row_index,'ML Score'] = np.round(metrics.roc_auc_score(y_test, y_pred),3)

    #Collect Mean Test scores for statistical significance test
    test_scores.append(grid_search.cv_results_['mean_test_score'])
    row_index+=1
```

3.8.4.15.3 Results table from the Model evaluation with metrics.

In [108]: MLA_compare.sort_values(by = ['Score'], ascending = False, inplace = True)
          MLA_compare

Out[108]:

	Name	Score	Accuracy_Score	ROC_AUC_score	final_rmse	Classification_error	Recall_Score	Precision_Score	mean_test_score	mean_fit_
3	XGBClassifier	1	1	1	0	0	1	1	0.815627	1
4	XGBClassifier_30%_Data	0.99	1	0.99	0.00511807	3e-05	0.97971	1	0.815627	1
5	XGBClassifier_All_Data	0.988	1	0.988	0.00557846	3e-05	0.97614	0.99975	0.815627	1
2	SVC	0.664	1	0.759	0.0205282	0.00042	0.51825	0.92208	0.372893	66
0	LogisticRegression	0.409	0.999	0.631	0.0246718	0.00061	0.26277	0.92308	0.226208	29
1	KerasClassifier	0.207	0.999	0.562	0.0275839	0.00076	0.12409	0.62963	0.0918313	5

Figure 29: scikit-learn

3.8.4.15.4 ROC AUC Score

AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.
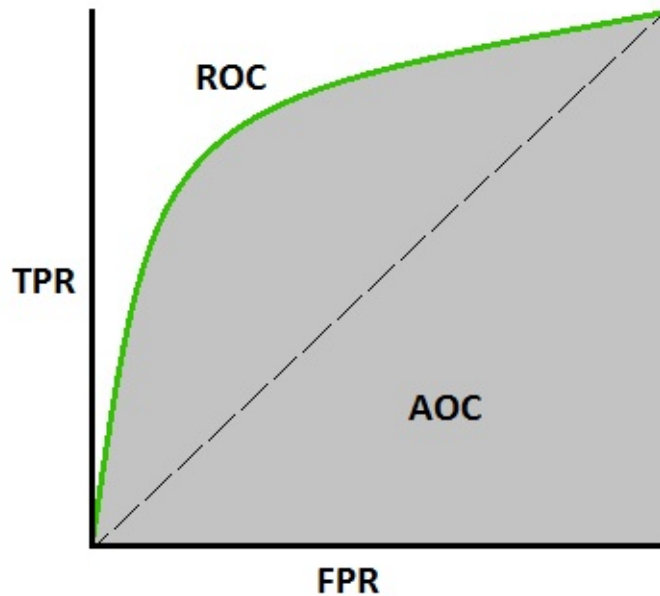
Figure 30: scikit-learn

```
In [109]: sns.barplot(x='ROC_AUC_score', y = 'Name', data = MLA_compare, color = 'm')

          plt.title('Machine Learning Algorithm ROC AUC Score \n')
          plt.xlabel('ROC_AUC Score (%)')
          plt.ylabel('Algorithm')

Out[109]: Text(0,0.5,'Algorithm')
```
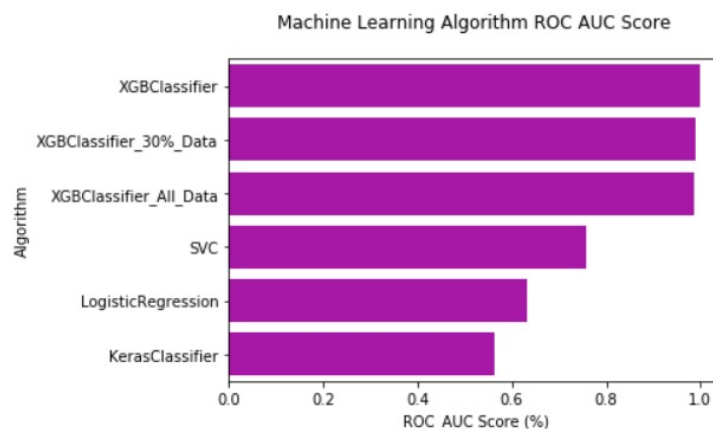
Figure 31: scikit-learn

3.8.4.16 K-means in scikit learn.

3.8.4.16.1 Import

3.8.4.17 K-means Algorithm

In this section we demonstrate how simple it is to use k-means in scikit learn.

3.8.4.17.1 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
```

3.8.4.17.2 Create samples

```
np.random.seed(42)

digits = load_digits()
data = scale(digits.data)
```

3.8.4.17.3 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')
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    %.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)

              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 * '_')
```

3.8.4.17.4 Visualize

See [Figure 32](#)

```
bench_k_means(KMeans(init=pca.components_,
                     n_clusters=n_digits, n_init=1),
              name="PCA-based",
              data=data)
print(79 * '_')
```

3.8.4.17.5 Visualize

See [Figure 32](#)

```
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()
```
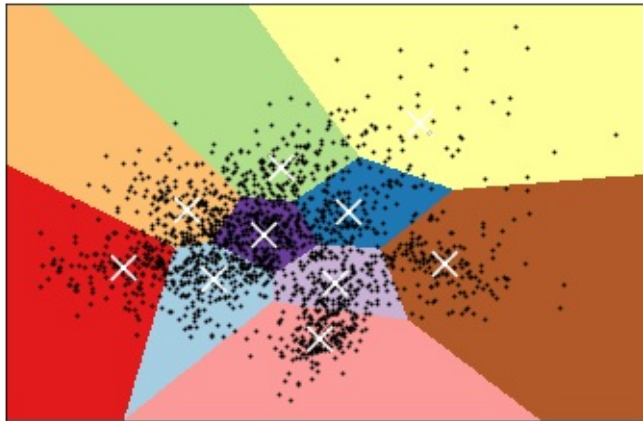

Figure 32: Result

3.8.5 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.

3.8.5.1 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.

3.8.5.1.1 `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:

```python
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,arg=(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.

3.8.5.1.2 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:

```python
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:

```
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:

```python
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.

3.8.5.2 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.

3.8.5.2.1 Process

Similar to `threading` module which was employing `thread` (aka `_thread`) under the hood, `multiprocessing` employs the `Process` class. Consider the following example:

```python
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 `Process`es 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!
```

3.8.5.2.2 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-blockin` on the other hand, the control returns immediately.

3.8.5.2.2.1 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.

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`:

```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)
    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.

3.8.5.2.3 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.

3.8.5.2.4 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:

```python
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`es 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 process**es**-toprocess**es** 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.

3.8.5.2.4.1 Value

The following example re-implements the broken example in the previous section. We fix the strange output, by using both `Lock` and `Value`:

```python
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`:

```python
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
```

3.8.6 Dask - Random Forest Feature Detection ☁

3.8.6.1 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.

3.8.6.2 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 will load our data. pandas makes it easy!

```python
# 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.

```python
# for red wines
red_df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlor
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000
mean	8.319637	0.527821	0.270976	2.538806	0.087467

std	1.741096	0.179060	0.194801	1.409928	0.047065
min	4.600000	0.120000	0.000000	0.900000	0.012000
25%	7.100000	0.390000	0.090000	1.900000	0.070000
50%	7.900000	0.520000	0.260000	2.200000	0.079000
75%	9.200000	0.640000	0.420000	2.600000	0.090000
max	15.900000	1.580000	1.000000	15.500000	0.611000

```
# for white wines
white_df.describe()
```

	fixed acidity	**volatile acidity**	**citric acid**	**residual sugar**	**chlor**
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000
mean	6.854788	0.278241	0.334192	6.391415	0.045772
std	0.843868	0.100795	0.121020	5.072058	0.021848
min	3.800000	0.080000	0.000000	0.600000	0.009000
25%	6.300000	0.210000	0.270000	1.700000	0.036000
50%	6.800000	0.260000	0.320000	5.200000	0.043000
75%	7.300000	0.320000	0.390000	9.900000	0.050000
max	14.200000	1.100000	1.660000	65.800000	0.346000

Sometimes it is easier to understand the data visually. A histogram of the white wine quality *data citric* acid samples is shown next. You can of course visualize other columns' data or other datasets. Just replace the DataFrame and column name (see [Figure 33](#)).

```
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()
```
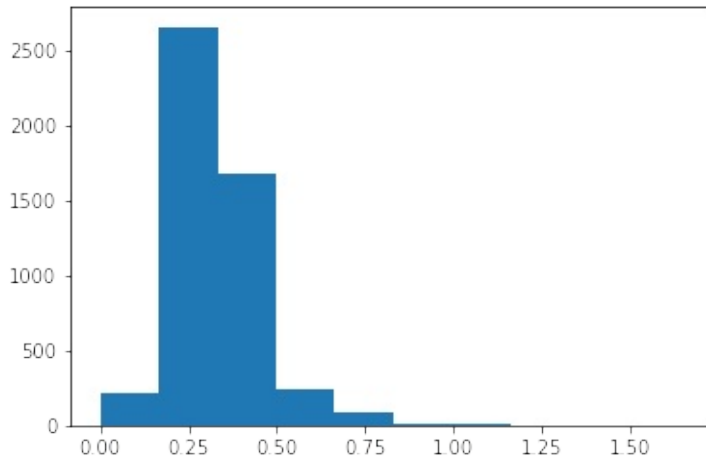
Figure 33: Histogram

3.8.6.3 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.

3.8.6.3.1 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 vlues 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.

```python
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.

3.8.6.4 Random Forest

```python
# pick a classifier

from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor,ExtraTreeClassifier,ExtraTreeRegressor
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 34

```
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()
```
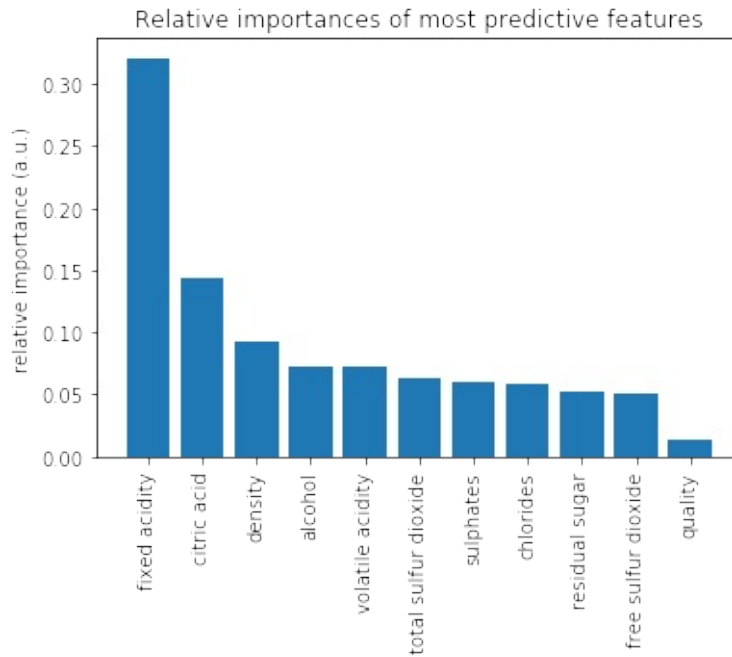
Figure 34: Result

```
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)
```

3.8.6.5 Acknowledgement

This notebook was developed by Juliette Zerick and Gregor von Laszewski

4 DEVOPS TOOLS

4.1 REFCARDS �',☁

🎓 Learning Objectives

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

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

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

CheatSheets

- [CheatSheets](#)

Editors

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

Documentation

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

Linux

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

Cloud/Virtualization

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

SQL

- [SQL](#)

Languages

- [R](#)

Python

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

4.2 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.

4.2.1 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.

4.2.2 Guest additions

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

- Resize the windows of the vm

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

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

A video is located at

 [Virtualbox (4:46)](#)

Please reboot the machine after installation and configuration.

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

OSX to Vbox:

    command c shift CONTRL v

Vbox to OSX:

    shift CONTRL v shift CONTRL v

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

4.2.3 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.*

4.3 VAGRANT

🎓 Learning Objectives

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

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

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

Detailed documentation for it is located

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

A list of *boxes* is available from

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

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

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

4.3.1 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

4.3.1.1 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`

4.3.1.2 Windows ⓞ

⊗ ⍰

⍰ students contribute

4.3.1.3 Linux ⓞ

⊗ ⍰

(?) students contribute

4.3.2 Usage

To download, start and login into install the 18.04:

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

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

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

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

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

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

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

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

```
alias python=python3
```

When you exit the virtual machine with the

```
exit command
```

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

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

to manage the vm.

4.4 LINUX SHELL ☁

🎓 **Learning Objectives**

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

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

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

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

4.4.1 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.

4.4.2 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:

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 als find even as a RefCard.

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

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

.

Command	Description
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 file*a*
cat -n *filename*	print content of file*a* 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
who	displays a list of users that are currently logged on, for each user the login name, date and time of login, tty name, and hostname if not local are displayed
whoami	displays the users effective id see also id
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 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	execute a command as another user
su	substitute user identity
uname	print the operating system name
set -o emacs	tells the shell to use Emacs commands.
chmod go-rwx *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 `.py` at the end
ps	list the running processes
kill -9 1234	kill the process with the id 1234
at	que commands for later execution
cron	daemon to execute scheduled commands
crontab	manage the time table for execution commands with cron
mount /dev/cdrom /mnt/cdrom	mount a filesystem from a cd rom to /mnt/cdrom
users	list the logged in users
who	display who is logged in
whoami	print the user id
dmesg	display the system message buffer
last	indicate last logins of users and ttys

uname	print operating system name
date	prints the current date and time
time *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
curl	transfer a URL
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

4.4.3 The command man

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

```
$ man ls
```

You will see somthing like this

```
LS(1)                    BSD General Commands Manual                    LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION

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

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

     The following options are available:

     -@      Display extended attribute keys and sizes in long (-l) output.

     -1      (The numeric digit ``one''.)  Force output to be one entry
             per line.  This is the default when output is not to a terminal.

     -A      List all entries except for . and ...  Always set for the
             super-user.

     -a      Include directory entries whose names begin with a dot (.).


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

4.4.4 Multi-command execution

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

To execute command 2 once command 1 has finished use

```
command1; command2
```

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

```
command1; command2
```

To execute command 1 in the background use

```
command1 &
```

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

4.4.6 bashrc, bash_profile or zprofile

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

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

4.4.7 Makefile

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

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

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

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

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

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

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

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

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

In addition we can define variables in a makefile such as

```
HELLO="Hello World"

hello:
    @echo $(HELLO)
```

and can use them in our text with $ invocations.

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

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

4.4.8 chmod

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

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

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

Permission:	Description:
r	Read
w	Write
x	Execute file or access directory
X	Execute only if the object is a directory
s	Set the user or group ID when running
t	Restricted deletion flag or sticky mode
u	Specifies the permissions the user who owns the file has
g	Specifies the permissions of the group
o	Specifies the permissions of users not in the group

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

```
$ chmod –v o+rw file.txt
```

Multiple files can also be specified:

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

4.4.9 Exercises

E.Linux.1

Familiarize yourself with the commands

E.Linux.2

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

E.Linux.3

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

E.Linux.4

Should there be other commands listed in the table with the Linux

commands If so which? Create a pull request for them.

E.Linux.5

Write a section explaining chmod. Use letters not numbers

E.Linux.6

Write a section explaining chown. Use letters not numbers

E.Linux.7

Write a section explaining su and sudo

E.Linux.8

Write a section explaining cron, at, and crontab

4.5 SECURE SHELL ❖

🎓 Learning Objectives

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

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

number of computers can connect to a server that hosts such a list. This mechanism builds the basis for networked computers.

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

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

4.5.1 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    |
|O.  = ......     |
```

```
| = .   . .       |
+-----------------+
```

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 AAAAB3NzaC1yc2EAAAADAQABAAABAQCXJH2iG2FMHqC6T/U7uB8kt
6KlRh4kUOjgw9sc4Uu+Uwe/kshuispauhfsjhfm,anf6787sjgdkjsgl+EwD0
thkoamyi0VvhTVZhj61pTdhyl1t8hlkoL19JVnVBPP5kIN3wVyNAJjYBrAUNW
4dXKXtmfkXp98T3OW4mxAtTH434MaT+QcPTcxims/hwsUeDAVKZY7UgZhEbiE
xxkejtnRBHTipi0W03W05TOUGRW7EuKf/4ftNVPilCO4DpfY44NFG1xPwHeim
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 se a list of files such as

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

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

```
ssh-keygen -p
```

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

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

4.5.2 ssh-add

Often you wil 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
`-c`	Causes a confirmation to be requested from the user every time the added identities are used for authentication. The confirmation is requested using ssh-askpass.
`-D`	Deletes all identities from the agent.
`-d`	Deletes the given identities from the agent. The private key files for the identities to be deleted should be listed on the command line.
`-e` pkcs11	Remove key provided by pkcs11
`-L`	Lists public key parameters of all identities currently represented by the agent.
`-l`	Lists fingerprints of all identities currently represented by the agent.
`-s` pkcs11	Add key provided by pkcs11.
`-t` life	Sets the maximum time the agent will keep the given key. After the timeout expires, the key will be automatically removed from the agent. The default value is in seconds, but can be suffixed for m for minutes, h for hours, d for days, or w for weeks.
`-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.

4.5.3 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 toe 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
```

4.5.3.1 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.

4.5.3.2 Using SSH on Linux

All Linux versions come with ssh and can be used right from the terminal.

4.5.3.3 Using SSH on Raspberry Pi 3/4

SSH is available on Raspbian. However, to ssh into the PI you have to activate it via the configuration menu.

4.5.3.4 Accessing a Remote Machine

Once the key pair is generated, you can use it to access a remote machine. To dod so the public key needs to be added to the `authorized_keys` file on the remote machine.

The easiest way to do tis 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
```

4.5.4 SSH Port Forwarding ⬛

⬛ TODO: Add images to illustrate the concepts

SSH Port forwarding (SSH tunneling) creates an encrypted secure connection between a local computer and a remote computer through which services can be relayed. Because the connection is encrypted, SSH tunneling is useful for transmitting information that uses an unencrypted protocol.

4.5.4.1 Prerequisites

- Before you begin, you need to check if forwarding is allowed on the SSH server you will connect to.
- You also need to have a SSH client on the computer you are working on.

If you are using the OpenSSH server:

```
$ vi /etc/ssh/sshd_config
```

and look and change the following:

```
AllowTcpForwarding = Yes
GatewayPorts = Yes
```

Set the `GatewaysPorts` 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.

4.5.4.2 How to Restart the Server

If you are on:

- Linux, depending upon the init system used by your distribution, run:

  ```
  $ sudo systemctl restart sshd
  $ sudo service sshd restart
  ```

  Note that depending on your distribution, you may have to change the service to ssh instead of sshd.

- Mac, you can restart the server using:

  ```
  $ sudo launchctl unload /System/Library/LaunchDaemons/ssh.plist
  $ sudo launchctl load -w /System/Library/LaunchDaemons/ssh.plist
  ```

- Windows and want to set up a SSH server, have a look at MSYS2 or Cygwin.

4.5.4.3 Types of Port Forwarding

There are three types of SSH Port forwarding:

4.5.4.4 Local Port Forwarding

Local port forwarding lets you connect from your local computer to another server. It allows you to forward traffic on a port of your local computer to the SSH server, which is forwarded to a destination server. To use local port forwarding, you need to know your destination server, and two port numbers.

Example 1:

```
$ ssh -L 8080:www.cloudcomputing.org:80 <host>
```

Where `<host>` should be replaced by the name of your laptop. The -L option specifies local port forwarding. For the duration of the SSH session, pointing your browser at `http://localhost:8080/` would send you to `http://cloudcomputing.com`

Example 2:

This example opens a connection to the www.cloudcomputing.com jump server, and forwards any connection to port 80 on the local machine to port 80 on `intra.example.com`.

```
$ ssh -L 80:intra.example.com:80 www.cloudcomputing.com
```

Example 3:

By default, anyone (even on different machines) can connect to the specified port on the SSH client machine. However, this can be restricted to programs on the same host by supplying a bind address:

```
$ ssh -L 127.0.0.1:80:intra.example.com:80 www.cloudcomputing.com
```

Example 4:

```
$ ssh -L 8080:www.Cloudcomputing.com:80 -L 12345:cloud.com:80 <host>
```

This would forward two connections, one to `www.cloudcomputing.com`, the other to `www.cloud.com`. Pointing your browser at `http://localhost:8080/` would download pages from www.cloudcomputing.com, and pointing your browser to `http://localhost:12345/` would download pages from www.cloud.com.

Example 5:

The destination server can even be the same as the SSH server.

```
$ ssh -L 5900:localhost:5900 <host>
```

The LocalForward option in the OpenSSH client configuration file can be used to configure forwarding without having to specify it on command line.

4.5.4.5 Remote Port Forwarding

Remote port forwarding is the exact opposite of local port forwarding. It forwards traffic coming to a port on your server to your local computer, and then it is sent to a destination. The first argument should be the remote port where traffic will be directed on the remote system. The second argument should be the address and port to point the traffic to when it arrives on the local system.

```
$ ssh -R 9000:localhost:3000 user@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 completing the previous steps you should be able to connect to the server remotely, even from your local machine. `ssh -R` first creates an SSH tunnel that forwards traffic from the server on port 9000 to your local machine on port 3000.

4.5.4.6 Dynamic Port Forwarding

Dynamic port forwarding turns your SSH client into a SOCKS proxy server. SOCKS is a little-known but widely-implemented protocol for programs to request any Internet connection through a proxy server. Each program that uses the proxy server needs to be configured specifically, and reconfigured when you stop using the proxy server.

```
$ ssh -D 5000 user@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.

4.5.4.7 ssh config

Defaults and other configurations can be added to a configuration file that is placed in the system. The ssh program on a host receives its configuration from

- the command line options
- a user-specific configuration file: `~/.ssh/config`
- a system-wide configuration file: `/etc/ssh/ssh_config`

Next we provide an example on how to use a config file

4.5.4.8 Tips

Use SSH keys

- You will need to use ssh keys to access remote machines

No blank passphrases

- In most cases you must use a passphrase with your key. In fact if we find that you use passwordless keys to futuresystems and to chameleon cloud resources, we may elect to give you an*F* for the assignment in question. There are some exceptions, but they will be clearly communicated to you in class. You will as part of your cloud drivers license test explain how you gain access to futuresystems and chameleon to explicitly explain this point and provide us with reasons what you can not do.

A key for each server

- Under no circumstances copy the same private key on multiple servers. This violates security best practices. Create for each server a new private

key and use their public keys to gain access to the appropriate server.

Use SSH agent

- So as to not to type in all the time the passphrase for a key, we recommend using ssh-agent to manage the login. This will be part of your cloud drivers license.

  But shut down the ssh-agent if not in use

keep an offline backup, put encrypt the drive

- You may for some of our projects need to make backups of private keys on other servers you set up. If you like to make a backup you can do so on a USB stick, but make sure that access to the stick is encrypted. Do not store anything else on that key and look it in a safe place. If you lose the stick, recreate all keys on all machines.

4.5.4.9 References

- [The Secure Shell: The Definitive Guide, 2 Ed (O'Reilly and Associates)](#)

4.5.5 SSH to FutureSystems Resources ☁

🎓 **Learning Objectives**

- Obtain a Future system account so you can use kubernetes or dockerswarm or other services offered by FutureSystems.

Next, you need to upload the key to the portal. You must be logged into the portal to do so.

Step 1: Log into the portal https://portal.futuresystems.org/

Step 2: Click on the "MY ACCOUNT" link.

Step 3: Click on "EDIT"

Step 4: Paste your ssh key into the box marked Public SSH Key. Use a text editor to open the `id_rsa.pub`. Copy the entire contents of this file into the ssh key field as part of your profile information. Many errors are introduced by users in this step as they do not paste and copy correctly.

If you need to add keys, use the Add another item button

At this point, you have uploaded your key. However, you will still need to wait till all accounts have been set up to use the key, or if you did not have an account till it has been created by an administrator. Please, check your email for further updates. You can also refresh this page and see if the boxes in your account status information are all green. Then you can continue.

4.5.5.1 Testing your FutureSystems ssh key

If you have had no FutureSystem account before, you need to wait for up to two business days so we can verify your identity and create the account. So please wait. Otherwise, testing your new key is almost instantaneous on india. For other clusters like it can take around 30 minutes to update the ssh keys.

To log into india simply type the usual ssh command such as:

```
$ ssh portalname@india.futuresystems.org
```

The first time you ssh into a machine you will see a message like this:

```
The authenticity of host 'india.futuresystems.org (192.165.148.5)' cannot be established.
RSA key fingerprint is 11:96:de:b7:21:eb:64:92:ab:de:e0:79:f3:fb:86:dd.
Are you sure you want to continue connecting (yes/no)? yes
```

You have to type yes and press enter. Then you will be logging into india. Other FutureSystem machines can be reached in the same fashion. Just replace the name india, with the appropriate FutureSystems resource name.

4.5.6 Exercises ☁

E.SSH.1:

Create an SSH key pair

E.SSH.2:

Upload the public key to git repository you use.

E.SSH.3:

What is the output of a key that has a passphrase when executing the following command. Test it out on your key

```
$ grep ENCRYPTED ~/.ssh/id_rsa
```

E.SSH.4

Get an account on futuresystems.org (if you are authorized to do so). Upload your key to https://futuresystems.org. Login to india.futuresystems.org. Note that this could take some time as administrators need to approve you. Be patient.

E.SSH.5:

What can happen if you copy your private key to a machine on the network?

E.SSH.6:

Should I share my provate key with others?

E.SSH.7:

Assume I participate in a video conference call and I accidently share my private key. What should I do?

E.SSH.8:

Assume I participate in a video conference call and I accidently share my public key. What should I do?

4.6 GITHUB ☁

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.

4.6.1 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.

4.6.2 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 (⬛ 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.

4.6.3 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

4.6.4 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 some times it is impossible to achieve what we just

described as conflicts may exist. For example, you and the to-be-rebased copy both edited some common text section. Once this happens, human intervention needs to take place to resolve the conflict.

[Git 4:20 Rebase](#)

4.6.5 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.

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

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

4.6.6 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](#)

4.6.7 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](#)

4.6.8 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](#)

4.6.9 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.

[Git 3:11 Merge](#)

4.6.10 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.

[Git 3:47 GUI](#)

There are many other git GUI tools available that directly integrate into your

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

4.6.11 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

[📹 Git 1:25 Windows](#)

4.6.12 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.

4.6.13 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**.

4.6.14 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.

4.6.15 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 god to still create a branch, work in the branch and than merge the branch into the master once you verified things work. After you merged you can push the content to the github repository.

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

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

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

Afterwards, commit it using:

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

Then push it to your remote repository using:

```
git push
```

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

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

afterwards:

```
git push
```

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

```
git pull
```

4.6.16 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 ay 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.

4.6.17 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 chose into which account you like to fork the original repository. Next you will create a colne from your corked directory. You will see in your fork a green clone button. You will see a URL that you can copy into your terminal. If the links does not include your username, it is the wrong link.

In your terminal you now say

```
git colne  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.

4.6.17.1 Stay up to date with the original repo

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

Make sure you have upstream repo defined:

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

Now Get latest from upstream:

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

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

```
$ git status
```

should show the name of the conflicting file:

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

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

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

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

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

At this stage, only refs.bib is changed:

```
$ git status
```

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

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

And finally push the last committed change:

```
$ git push
```

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

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

4.6.17.2 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

4.6.18 Exercises

E.Github.1:

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

E.Github.2:

What is the differencebetween 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.*

4.6.19 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 conduction 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 toa 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/

4.6.19.1 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.

4.6.19.2 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.

4.6.19.2.1 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 ot the task text, you need to escape it with a `\`

4.6.19.2.2 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/

4.6.19.2.3 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.

4.6.19.2.4 Emojis

Although github supports emojis such as `:+1:` we do not use them typically in our class.

4.6.19.3 Notifications

Github allows you to set preferences on how you lik 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.

4.6.19.4 cc

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

4.6.19.5 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.

4.6.20 Glossary

The Glossary is copied from

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

Add
> put a file (or particular changes thereto) into the index ready for a commit operation. Optional for modifications to tracked files; mandatory for hitherto un-tracked files.

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

Commit

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

Commit object

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

Fast-forward

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

Fetch

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

HEAD

the latest state of the current branch.

Index

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

Master

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

Merge

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

Object

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

Origin

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

Pull

shorthand for a fetch followed by a merge (or rebase if –rebase option is used).

Push

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

Rebase

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

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

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

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

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

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

Un-tracked
: not known currently to git.

4.6.21 Example commands

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

The command list is copied from

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

4.6.21.1 Local commands to version contril your files

Obtain differences with

```
$ git status
```

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

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

Delete unwanted tracked files:

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

Add un-tracked files:

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

Stage a modified file for commit:

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

Commit currently-staged files:

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

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

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

Commit all modified files:

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

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

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

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

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

Get differences between local file and committed version:

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

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

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

Change to an existing local branch:

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

Merge another branch into the current one:

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

4.6.21.2 Interacting with the remote

Get the current list of remotes (including URIs) with

```
$ git remote -v
```

Get the current list of defined branches with

```
$ git branch -a
```

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

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

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

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

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

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

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

```
$ git pull
```

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

```
$ git cherry -v
```

Push changes to the remote tracking branch:

```
$ git push
```

Push all changes to all tracking branches:

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

4.7 GIT PULL REQUEST ☁

4.7.1 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.

4.7.2 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.

4.7.3 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 `YOURGITUSERANAME` is indeed your github user name. That is because you are now in your own copy of the `cloudmesh/technologies` repository. n our case the user name will be `pulashti`.

4.7.4 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.

4.7.5 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)
```

4.7.6 Making changes

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

```
$ git pull upstream master
```

Now make the needed changes commit and push, the changes will be pushed to

your copy of the repo i Github, not the `cloudmesh/technologies` repo.

4.7.7 Creating a pull request

Once we have changes pushed, you can go into your repository in Github to create a pull request. As seen in @#fig:button-pullrequest, you have an button named `Pull request`
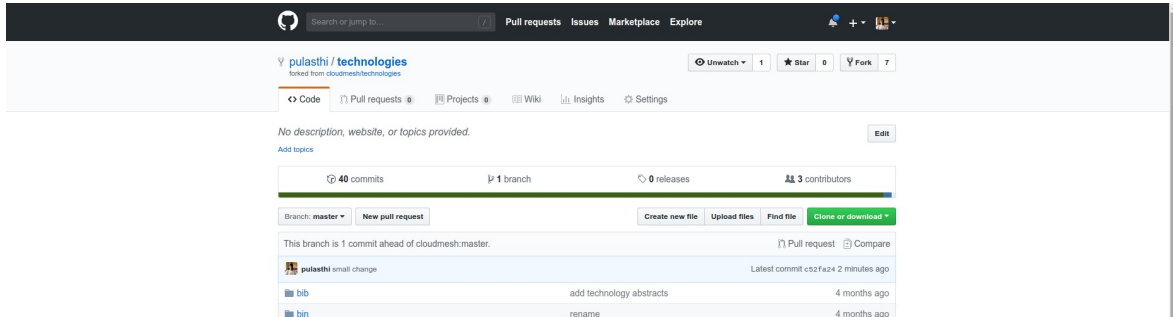
Figure 35: Button Pull request

Once you click on that button you will be taken to a page to create the pull request, which will look similar to [Figure 36](#).
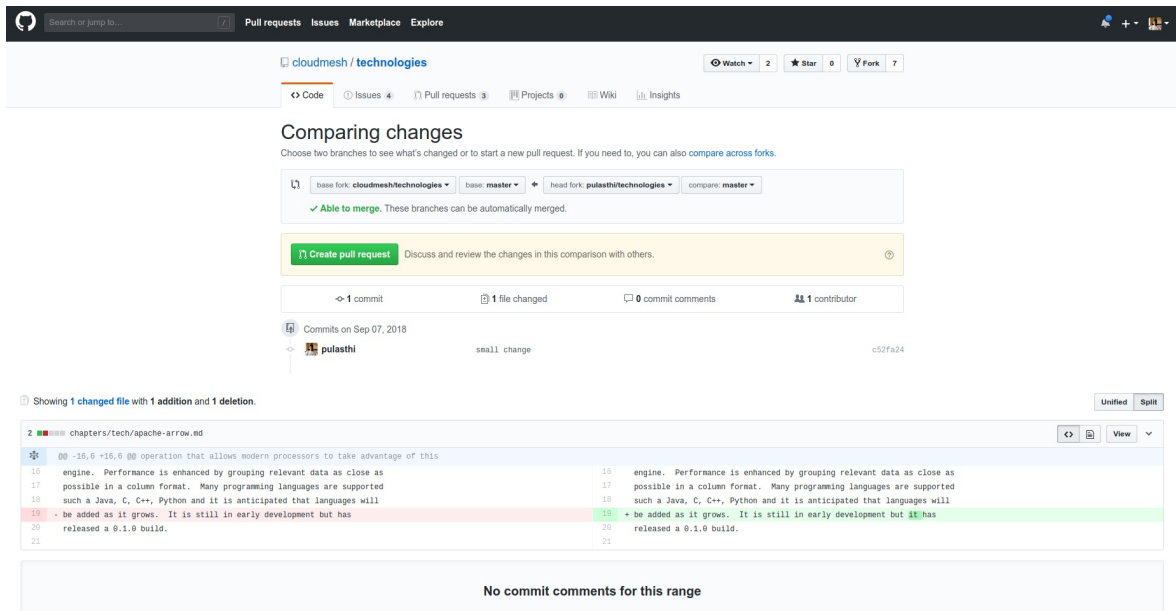
Figure 36: Create a pull request

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

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

 git example on CL 10:09

4.8 TIG

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

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

On OSX it can be installed with:

```
$ brew install tig
```

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

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

A screenshot shows some if its basic functionality is shown in [Figure 37](#)

Figure 37: Git tig main vie

Example infocations are

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

5 Introduction to Cloud Computing and Data Engineering for Cloud Computing and Machine Learning ☁

E222 Intelligent Systems II and E516 Engineering Cloud Computing

YouTube Playlist https://www.youtube.com/playlist?list=PLy0VLh_GFyz81ZFQ6Xrd1PHHI1EzjhhVb

PowerPoint of full set A) to U) https://drive.google.com/open?id=1RQ8Q_A32ks02CSCZAzKiJJ9P9YntMRAo

5.1 A. Summary of Introduction to Cloud Computing & Data Engineering

This lesson summarizes the component lessons CloudIntroB to CloudIntroU of Introduction to Cloud Computing and Data Engineering Lecture

 Summary Cloud Computing 15:40

5.2 B. Defining Clouds I

- Basic definition of cloud and two very simple examples of why virtualization is important.
- How clouds are situated wrt HPC and supercomputers
- Why multicore chips are important
- Typical data center

 Defining Clouds I 20:22

5.3 C. Defining Clouds II

- Service-oriented architectures: Software services as Message-linked

computing capabilities
- The different aaS's: Network, Infrastructure, Platform, Software
- The amazing services that Amazon AWS and Microsoft Azure have
- Initial Gartner comments on clouds (they are now the norm) and evolution of servers; serverless and microservices
- 2016/2018 Infrastructure Strategies Hype Cycle and Priority Matrix

[Defining Clouds II 24:22](#)

5.4 D. DEFINING CLOUDS III

- Cloud Market Share
- How important are they?
- How much money do they make?

[Defining Clouds III 12:23](#)

5.5 E. VIRTUALIZATION

- Virtualization Technologies, Hypervisors and the different approaches

- KVM Xen, Docker and Openstack

- See:

  - https://en.wikipedia.org/wiki/Hypervisor
  - https://en.wikipedia.org/wiki/Xen
  - https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine
  - https://en.wikipedia.org/wiki/Operating-system-level_virtualization
  - https://medium.com/\@dbclin/aws-just-announced-a-move-from-xen-towards-kvm-so-what-is-kvm-2091f123991
  - https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers
  - https://en.wikipedia.org/wiki/OpenStack

[Virtualization 11:21](#)

5.6 F. Technology Hypecycle I

- Gartner's Hypecycles and especially that for emerging technologies in 2018, 2017 and 2016
- The phases of hypecycles
- Priority Matrix with benefits and adoption time
- Today clouds have got through the cycle (they have emerged) but features like blockchain, serverless and machine learning are on cycle
- Hypecycle and Priority Matrix for Data Center Infrastructure 2017 and 2018

[Technology Hypecycle I 31:23](#)

5.7 G. Technology Hypecycle II

- Emerging Technologies hypecycles and Priority matrix at selected times 2008-2015
- Clouds star from 2008 to today
- They are mixed up with transformational and disruptive changes
- The route to Digital Business (2015)

[Technology Hypecycle II 16:05](#)

5.8 H. Cloud Infrastructure I

- Comments on trends in the data center and its technologies
- Clouds physically across the world
- Green computing
- Fraction of world's computing ecosystem in clouds and associated sizes

[Cloud Infrastructure I 21:20](#)

5.9 I. Cloud Infrastructure II

Gartner hypecycle and priority matrix on Infrastructure Strategies and Compute Infrastructure
Containers compared to virtual machines
The emergence of artificial intelligence as a dominant force

[Cloud Infrastructure II 17:52](#)

5.10 J CLOUD SOFTWARE

- HPC-ABDS with over 350 software packages and how to use each of 21 layers
- Google's software innovations
- MapReduce in pictures
- Cloud and HPC software stacks compared
- Components need to support cloud/distributed system programming
- Single Program/Instruction Multiple Data SIMD SPMD

[Cloud Software 37:56](#)

5.11 K. CLOUD APPLICATIONS I

- Big Data; a lot of best examples have NOT been updated so some slides old but still make the correct points
- Some of the business usage patterns from NIST

[Cloud Applications I 11:58](#)

5.12 L CLOUD APPLICATIONS II

- Clouds in science where area called cyberinfrastructure; the usage pattern from NIST
- Artificial Intelligence from Gartner

[Cloud Applications II 13:03](#)

5.13 M Cloud Applications III

- Characterize Applications using NIST approach
- Internet of Things
- Different types of MapReduce

📹 [Cloud Applications III 24:12](#)

5.14 N. Clouds and Parallel Computing

- Parallel Computing in general
- Big Data and Simulations Compared
- What is hard to do?

[:clapper :Clouds and Parallel Computing 35:03](#)

5.15 O. Storage

- Cloud data approaches
- Repositories, File Systems, Data lakes

📹 [Storage 19:22](#)

5.16 P. HPC and Clouds

- The Branscomb Pyramid
- Supercomputers versus clouds
- Science Computing Environments

📹 [HPC and Clouds 19:29](#)

5.17 Q. Comparison of Data Analytics with Simulation

- Structure of different applications for simulations and Big Data
- Software implications
- Languages

📹 [Comparison of Data Analytics with Simulation 16:19](#)

5.18 R. JOBS

- Computer Engineering
- Clouds
- Design
- Data Science/Engineering

📹 [Jobs 15:30](#)

5.19 S. THE FUTURE I

- Gartner cloud computing hypecycle and priority matrix
- Hyperscale computing
- Serverless and FaaS
- Cloud Native
- Microservices

📹 [The Future I 29:29](#)

5.20 T. THE FUTURE AND OTHER ISSUES II

- Security
- Blockchain

📹 [The Future and other Issues II 11:30](#)

5.21 U. THE FUTURE AND OTHER ISSUES III

- Fault Tolerance

📹 [The Future and other Issues III 9:10](#)

6 REST

6.1 INTRODUCTION TO REST

Learning Objectives

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

REST stands for **RE**presentational **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 Roy T. Fielding](#) [40].

Hence REST can use HTTP for the four CRUD operations:

- **C**reate resources
- **R**ead resources
- **U**pdate resources
- **D**elete resources

As part of the HTTP protocol we have methods such as GET, PUT, POST, and DELETE. These methods can than 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 [REST on Wikipedia](#) [41]).

6.1.0.1 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.

6.1.0.2 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.

6.1.0.3 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 Specification Frameworks:

These are frameworks that help defining rest servicice through specifications to generate REST services in a language and framework independent way. This includes for example Swagger 2.0 [42], OpenAPI 3.0 [43], and RAML [44].

REST programming language support:

These tools and services are targeting a particular programming language. Such tools include Flask Rest [45], and Django Rest Services [46], some of which we will explore in more detail.

REST documentation based tools:

These tools are primarily focusing on documenting REST specifications. Such tools include Swagger [47], 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 [47] as additional tools are available that can generate code from OpenAPI specifications [48], which we will explore in more detail.

A list of such efforts is available at OpenAPI Tools [49]

6.2 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.

6.2.1 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

Swagger Hub

> is a Web service to collaborativly develop and host OpenAPI specifications
> https://swagger.io/tools/swaggerhub/

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.

6.2.2 Swagger Community Tools

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

6.2.2.1 Converting Json Examples to OpenAPI YAML Models

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

- https://swagger-toolbox.firebaseapp.com/

The source code to this tool is available on github at

- https://github.com/essuraj/swagger-toolbox

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.

6.3 OPENAPI 2.0 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 wo 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.

6.3.1 The Virtual Cluster example API Definition

6.3.1.1 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.

6.3.1.2 Specification

```yaml
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
```

6.3.2 References

[The official OpenAPI 2.0 Documentation](#)

6.4 OPENAPI 3.0 REST SERVICE VIA INTROSPECTION ☁

The simplest way to create an OpenAPI service is to use the conexion service and read in the specification from its yaml file. It will than 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/book/tree/master/examples/rest/cpu

An extensive documentation is available at

- https://media.readthedocs.org/pdf/connexion/latest/connexion.pdf

This example will return dynamically the cpu information of a computer to demonstrate how simple it is to generate in python a REST service from an OpenAPI specification.

Our requirements.txt file includes

```
flask
connexion[swagger-ui]
```

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

```python
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

```yaml
openapi: 3.0.2
info:
  title: cpuinfo
  description: A simple service to get cpuinfo as an example of using OpenAPI 3.0
  license:
    name: Apache 2.0
  version: 0.0.1

servers:
 - url: http://localhost:8080/cloudmesh

paths:
  /cpu:
    get:
      summary: Returns cpu information of the hosting server
      operationId: cpu.get_processor_name
      responses:
        '200':
          description: cpu info
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/cpu"

components:
  schemas:
    cpu:
      type: "object"
      required:
        - "model"
      properties:
        model:
          type: "string"
```

Here we simply implement a get method and associate is 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

```python
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

```makefile
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.

6.4.1 Verification

It is important to be able to verify if a yaml file is correct. To identify this, the

easiest method is to use the swagger editor. There is an online verion available at:

- https://editor.swagger.io/

Go to the Web site, remove the current petstore example and simply paste your yaml file in it. Debug meessages will be helping you to correct things.

A terminal based command may als be helpful, but is a bit difficult to read.

```
$ connexion run cpu.yaml --stub --debug
```

6.4.2 Swagger-UI

Swagger comes with a convenient UI to invoke REST API calls using the web browser rather than relying on the curl commands.

Once the request and response definitions are properly specified, you can start the server by,

```
$ python server.py
```

Then the UI would also be spawned under the service URL *http://[service url]/ui/*

Example: http://localhost:8080/cloudmesh/ui/

6.4.3 Mock service

In some cases it may be useful to develop the API without having yet developed methods that you call with the OperationI. In this case it is useful to run a mock service. YOu can invoce such a service with

```
$ connexion run cpu.yaml --mock=all -v
```

6.4.4 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.

OpenAPI.Conexion.4:

Develop instructions on how to integrate the OpenAPI service framework in a WSGI based Web service. Chose a service you like so that the service could run in production.

OpenAPI.Conexion.5:

Develop instructions on how to integrate the OpenAPI service

framework in Tornado so the service could run in production.

6.5 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 is. 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.

6.5.1 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:

```yaml
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 specification and codegen"
  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"
```

6.5.2 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.

6.5.2.1 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/swagger-codegen-cli-2.
$ 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.

6.5.2.2 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.

6.5.2.3 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:

```python
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:

```python
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:

```python
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.

6.5.3 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.

6.5.3.1 Start a virtualenv:

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

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

6.5.3.2 Make sure you have the latest pip:

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

6.5.3.3 Install the requirements of the server side code:

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

6.5.3.4 Install the server side code package:

Under the same directory, run:

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

6.5.3.5 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)
```

6.5.3.6 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?

6.5.4 Step 4: Generate Client Side Code and Verify

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

6.5.4.1 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
```

6.5.4.2 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 ane 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
```

6.5.4.3 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:

```python
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.

6.5.5 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.

6.6 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 than 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.

```python
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'''
```

```python
    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)
```

6.7 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.

6.7.1 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
```

6.7.2 macOS install of MongoDB

On macOS you can use the command

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

6.7.3 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*.

6.7.4 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

6.7.5 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

6.7.6 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:

```python
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.

6.7.7 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 listed next, if your output is not formatted like this 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/2.7.16
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
    }
}
```

6.7.8 Creating REST API Endpoints

Next we wont 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**

```python
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': 'Zweistein',
        '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 lok. The route specifies how we access it from the service.

6.7.9 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.

6.7.10 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")
```

```python
    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.

6.7.11 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/cloudmesh/admin

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.

6.8 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. HATEOS 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 follwoing information when resources are created as showcased ot

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

Field	Description
_created	item creation date.
_updated	item last updated on.
_etag	ETag, to be used for concurrency control and conditional requests.
_id	unique item key, also needed to access the individual item endpoint.

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

6.8.1 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"
```

6.8.2 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
```

6.8.3 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
```

6.9 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 pic 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.

6.9.1 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.

6.9.1.1 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
```

6.9.1.2 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 se a number of json text been written to the screen.

6.9.1.3 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 evegenie to install it onto your system.

The original documentation for evegenie is located at:

- http://evegenie.readthedocs.io/en/latest/

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

- https://github.com/cloudmesh/evegenie

You clone it and install on your system as follows:

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

This should install in your system evegenie. YOu can verify this by typing:

```
$ which evegenie
```

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

```
$ evegenie

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.

6.10 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 aggording to the developers of that module:

1. *The Web browsable API improves usability.*
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."*

- https://www.django-rest-framework.org/

En example is provided on their Web Page at

- https://www.django-rest-framework.org/#example

To document your django framework with Swagger you can look at this example:

- https://www.django-rest-framework.org/topics/documenting-your-api/

However, we believe that for our purposes the approach to use conexion from an OpenAPI is much more appealing, also using conexion and also flask for the REST service is easier to acomplish. Django is a large package that will take mor time to getting used to.

6.11 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 the 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.

6.11.1 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 retrive 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 than use a program to convert LaTeX to markdown. For the later we can reuse a program called `pandoc` that can convert the table for 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 listes 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 that 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{/name}",
        "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.

6.11.2 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. How 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 data and visualize them graphically for one committer and all committers. Use bokeah or matplotlib.*

E.github.issues.6:

> *Develop a python program that retrieves a file. Deevlop a python program that uploads a file. Develop a class that does this and use it in your proggram. Use docopt to create a manual page. Please remember this prepares you for your project so this is very useful to do.*

7 MAPREDUCE

7.1 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.

7.1.1 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 38](#)):

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.
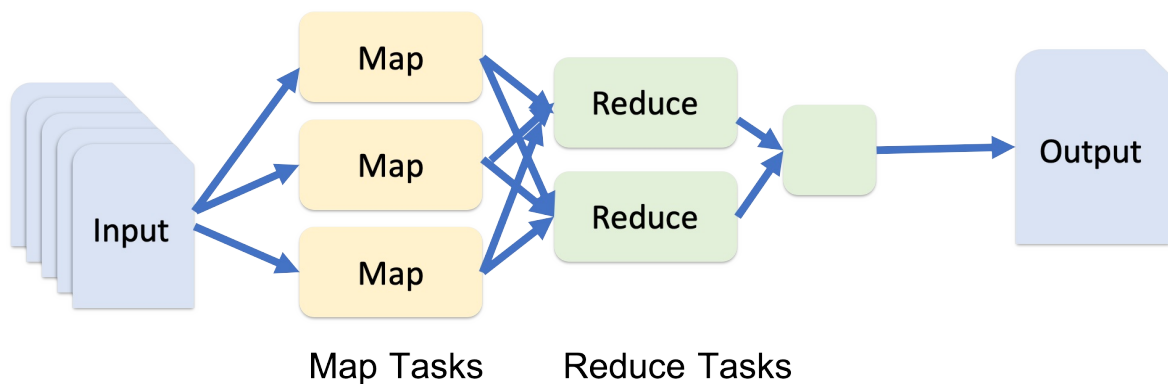
Figure 38: MapReduce Conceptual diagram

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 39)
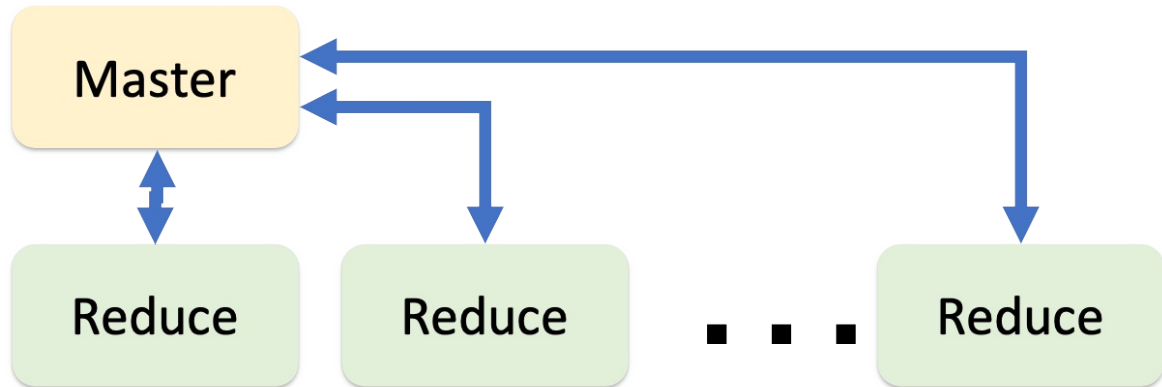
Figure 39: MapReduce Master worker diagram

7.1.1.1 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 40)
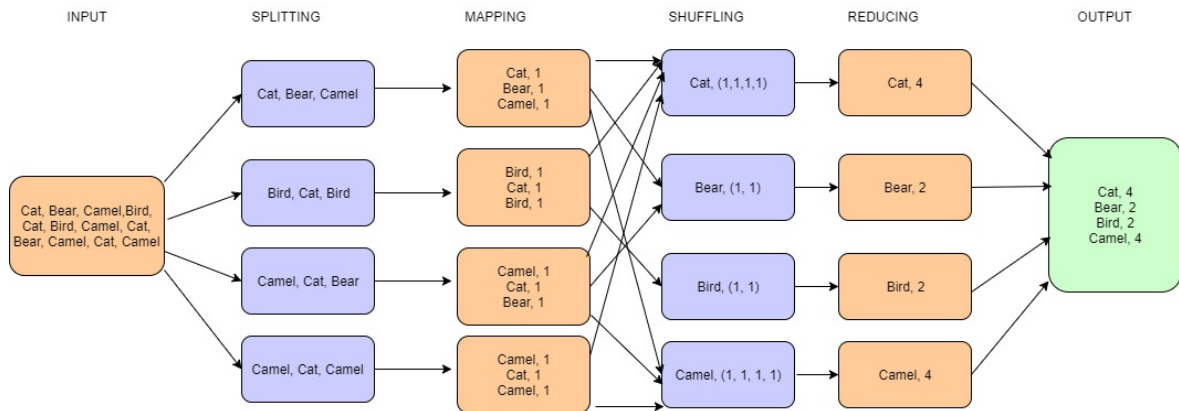
Figure 40: MapReduce WordCount [50]

Let us see an example of map() and reduce() methods in code for this word count example.

```java
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 next and both input and output here is a key/value pairs:

```java
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));
    }
}
```

7.1.2 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 41)
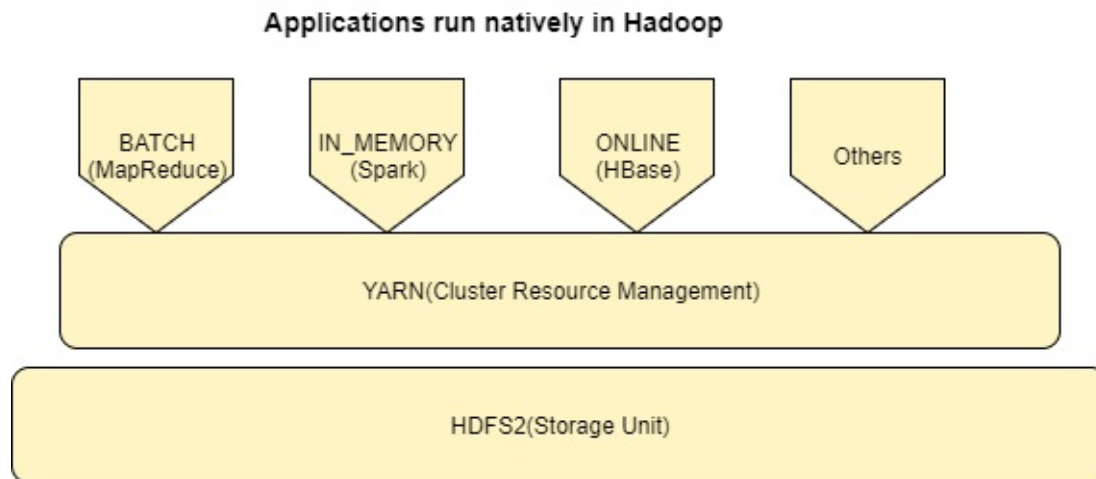
Figure 41: MapReduce Hadoop and Spark [51]

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.

7.1.2.1 Apache Spark

Apache Spark is lightning fast cluster computing framework. Spark is in-memory system. Spark is 100 time faster than Hadoop MapReduce.

7.1.2.2 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.

7.1.2.3 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.

7.1.3 References

- [52] https://www.ibm.com/analytics/hadoop/mapreduce
- [53] https://en.wikipedia.org/wiki/MapReduce
- [54] https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm
- [50] https://www.edureka.co/blog/mapreduce-tutorial/?utm_source=youtube&utm_campaign=mapreduce-tutorial-161216-wr&utm_medium=description
- [55] https://www.quora.com/What-is-the-difference-between-Hadoop-and-Spark
- [56] https://data-flair.training/blogs/apache-spark-vs-hadoop-mapreduce
- [51] https://www.youtube.com/watch?v=SqvAaB3vK8U&list=WL&index=25&t=2547s

7.2 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

7.2.1 Hadoop and MapReduce

In this section we discuss about the usage Hadoop MapReduce architecture.

[Hadoop 13:19 Hadoop B](#)

7.2.2 Hadoop EcoSystem

In this section we discuss about the Hadoop EcoSystem and the architecture.

[Hadoop 12:57 Hadoop C](#)

7.2.3 Hadoop Components

In this section we discuss about Hadoop Components in detail.

[Hadoop 15:14 Hadoop D](#)

7.2.4 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](#)

7.2.5 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 25:41 Hadoop F](#)

7.3 INSTALLATION OF HADOOP �".

This section is using Hadoop version 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.

If you use a newer version, and like to update this text please help

7.3.1 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.

7.3.2 Prerequisites

```
sudo apt-get install ssh
sudo apt-get install rsync
sudo apt-get install emacs
```

7.3.3 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`.

7.3.4 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:
SHA256:0UBCPd6oYp7MEzCpOhMhNiJyQo6PaPCDuOT48xUDDc0 hduser@computer
The key's randomart image is:
+---[RSA 2048]----+
|     .+ooo       |
| .    oE.oo      |
|+  .. ...+.      |
|X+=  .  o..      |
|XX.o  o.S        |
|Bo+ + .o         |
|*o * +.          |
|*.. *.           |
| +.o..           |
+----[SHA256]-----+
```

You have successfully configured ssh.

7.3.5 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.

7.3.6 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
```

7.3.7 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 next.

```
   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
                          Hadoop jar and the required libraries
    daemonlog             get/set the log level for each daemon
    trace                 view and modify Hadoop tracing settings
   or
    CLASSNAME             run the class named CLASSNAME

   Most commands print help when invoked w/o parameters.
```

7.4 HADOOP VIRTUAL CLUSTER INSTALLATION USING CLOUDMESH 〇 ☁

No

⚠ *This version is dependent on an older version of cloudmesh.*

✖ :TODO: we need to add the instalation instructions based on this version

7.4.1 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.

7.4.1.1 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.

7.4.1.2 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
xl41-001 129.114.33.147
xl41-002 129.114.33.148
xl41-003 129.114.33.149
```

If option `--no-floating-ip` is included during definition, you will see nodes without floating IP:

```
$ cm cluster nodes cluster-002
xl41-004 None
xl41-005 None
xl41-006 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 xl41-006
$ cm cluster nodes cluster-002
xl41-004 None
xl41-005 None
xl41-006 129.114.33.150
```

Then you can log in this node as a head node of your cluster by `cm vm ssh [NAME]`:

```
$ cm vm ssh xl41-006
cc@xl41-006 $
```

7.4.1.3 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
```

7.4.2 Hadoop Cluster Installation

This section is built upon the previous one. Please finish the previous one before start this one.

7.4.2.1 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.

7.4.2.2 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.

7.4.3 Advanced Topics with Hadoop

7.4.3.1 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
```

7.4.3.2 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:

```python
import sys

from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

  # tak 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)
...
```

7.5 SPARK

7.5.1 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.

7.5.1.1 Motivation for Spark

In this section we discuss about the background of Spark and core components of Spark.

📹 [Spark 15:57 Spark A](#)

7.5.1.2 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](#)

7.5.1.3 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](#)

7.5.1.4 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](#)

7.5.2 Installation of Spark ☁

In this section we will discuss how to install Spark 2.3.2 in Ubuntu 18.04.

7.5.2.1 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
```

7.5.2.2 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-pub/java/jdk/8u161-b1
tar xvzf 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
```

7.5.2.3 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
```

7.5.2.4 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
```

7.5.2.5 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'.
```
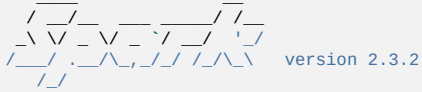
```
   Welcome to
        ____              __
       / __/__  ___ _____/ /__
      _\ \/ _ \/ _ `/ __/  '_/
     /___/ .__/\_,_/_/ /_/\_\   version 2.3.2
        /_/

   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.

7.5.2.6 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
```

7.5.2.7 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-evn.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

7.5.2.8 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(newLevel).
    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

7.5.3 Spark Streaming 🌩

7.5.3.1 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.

7.5.3.2 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.

```python
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 -lk 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.

7.5.3.3 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.

7.5.3.3.1 Step 1
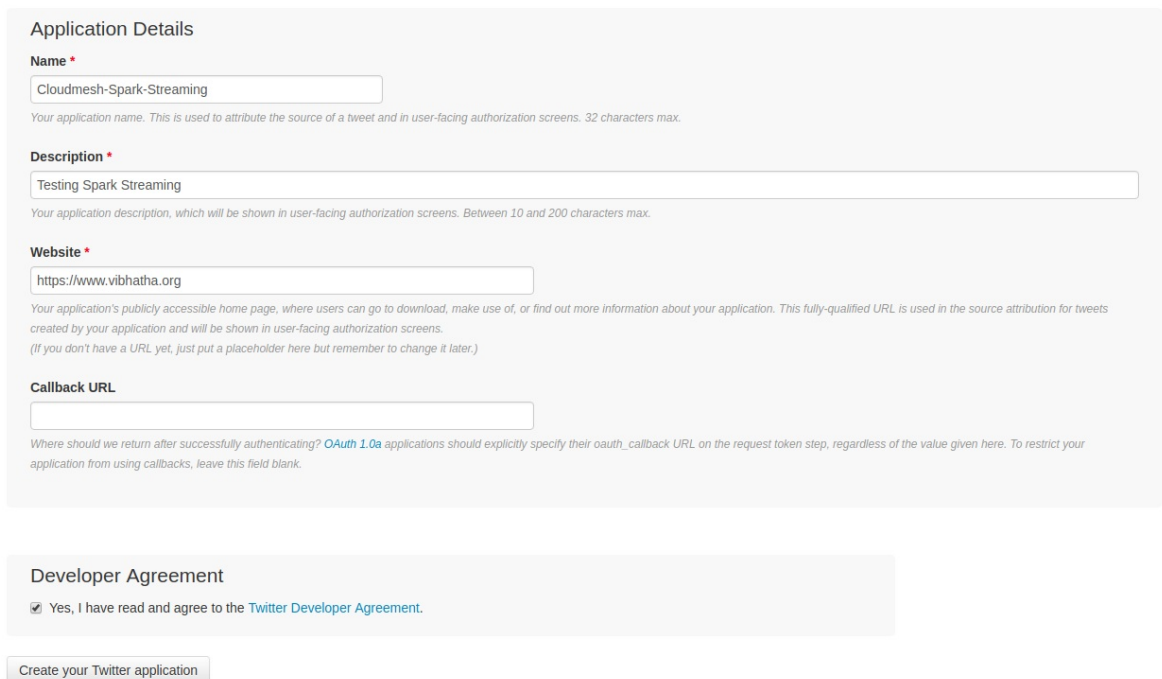
```
sudo pip install tweepy
```

7.5.3.3.2 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 42.

Figure 42: Create Twitter App

Next we need to to take a look at the dashboard created for the app. You can see how your dashboard looks like in +Figure 43.
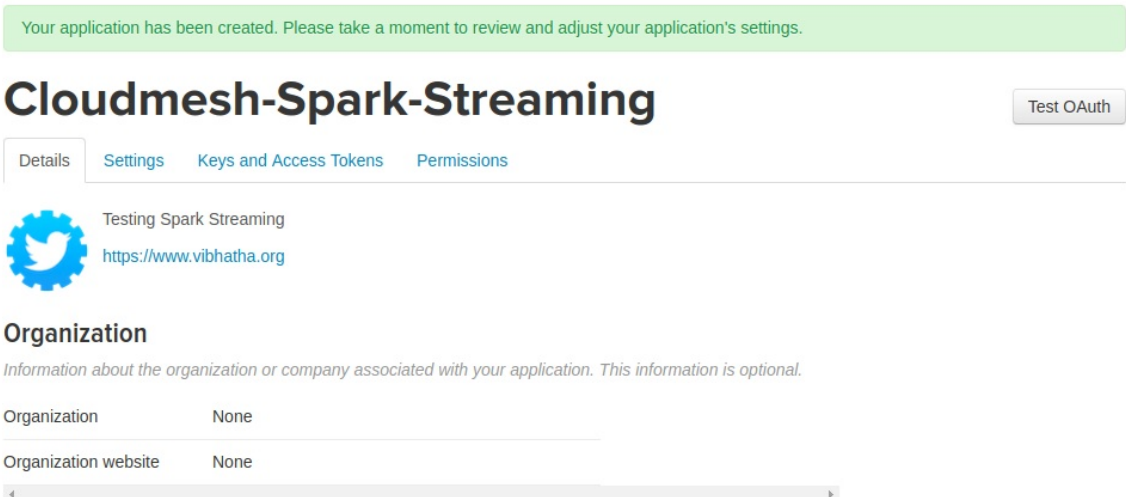
Figure 43: Go To Twitter App Dashboard

Next the application tokens generated must be reviewed and it can be found in +[Figure 44](#), you need to go to the `Keys and Access Tokens` tab.

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 44: 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 45

Figure 45: Create Your Twitter Access Tokens

The access tokens and keys are blurred in this section for privacy issues.

7.5.3.3.3 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
```

7.5.3.3.4 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.

```python
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 )
```

7.5.3.3.5 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 depending on your entity


    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 name according to your entity
      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()
```

7.5.3.3.6 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 46, +Figure 47, +Figure 48, +Figure 49.
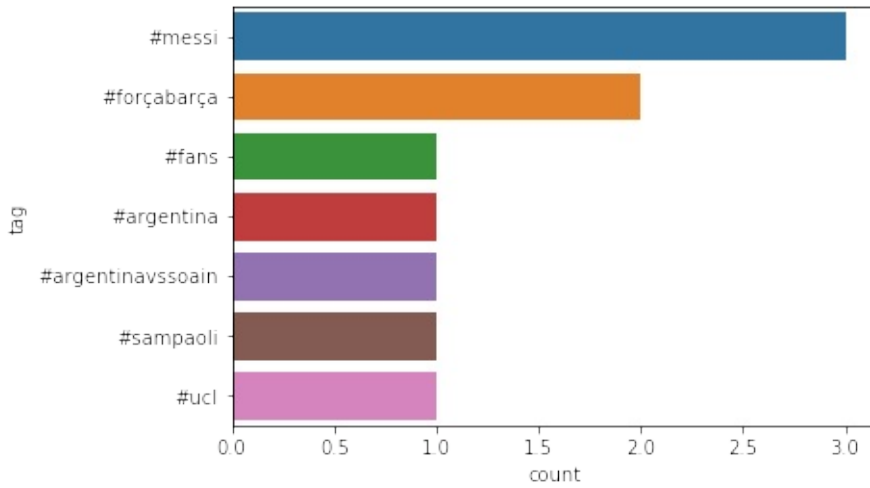
Figure 46: Twitter Topic Messi
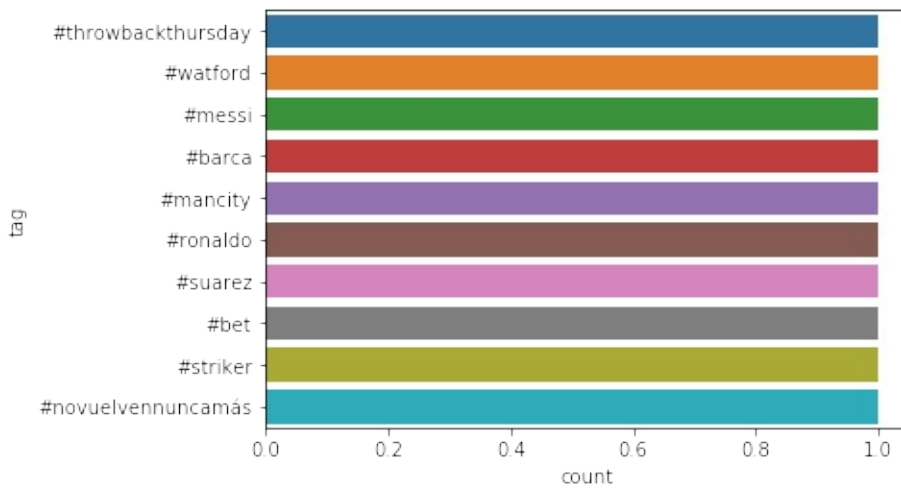
Figure 47: Twitter Topic Messi
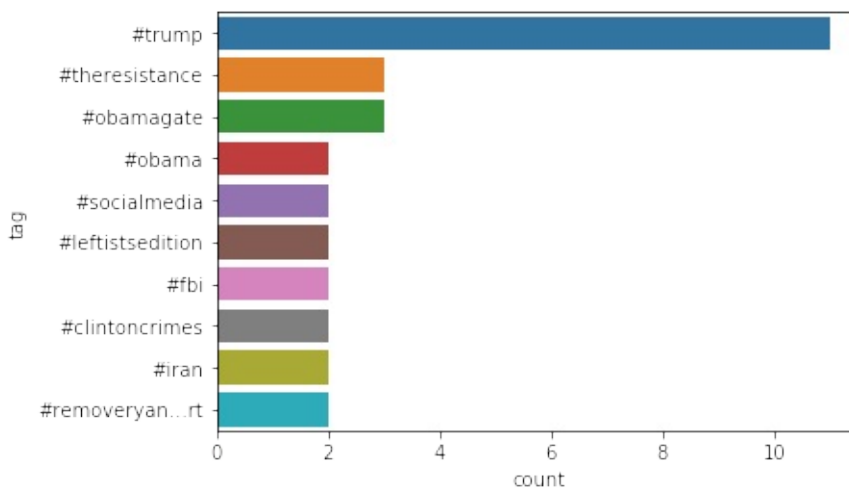
Figure 48: Twitter Topic Messi
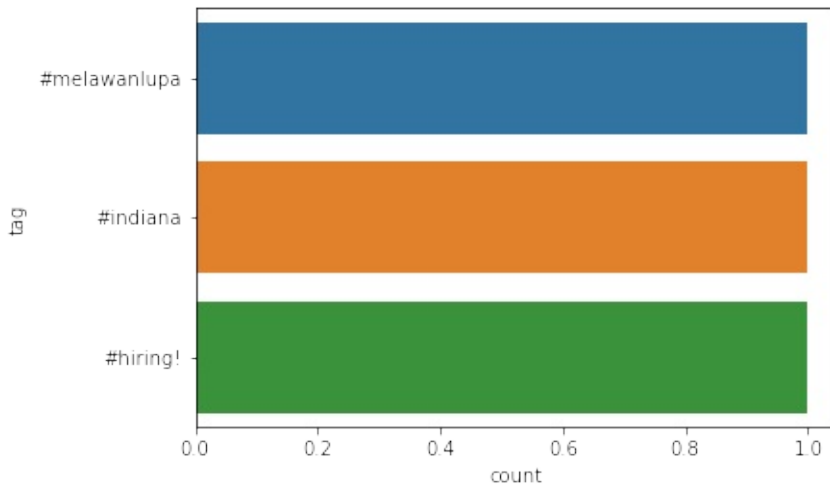
Figure 49: Twitter Topic Messi

7.5.4 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.

7.5.4.1 Resources

- https://spark.apache.org/
- http://www.scala-lang.org/
- https://docs.databricks.com/spark/latest/spark-sql/udf-in-python.html

7.5.4.2 Instructions for Spark installation

7.5.4.2.1 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/index.html

Second, setup environment variables for JDK by adding bin folder path to 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
```

7.5.4.3 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.html

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
```

7.5.4.4 MacOS

First, JDK should be installed to a path where there is no space in that path. Recommanded JAVA version is 8.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Second, setup environment variables for jdk by addding bin folder path to 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 varibale 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
```

7.5.4.5 Instructions for creating Spark User Defined Functions

7.5.4.5.1 Example: Temperature conversion

In this example we convert temperature data from Celsius to Fahrenheit with filtering and sorting

7.5.4.5.1.1 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)

7.5.4.5.1.2 How to write a python program with UDF

First, we need to import the relevent 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 colud 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()
```

7.5.4.5.1.3 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
```

7.5.4.5.1.4 Filtering and sorting

Now we are trying to find what is the maximum temperature reported for a particluar whether station and print the data in ascending order. We can achieve this by using **where()** and **orderBy()** fundtions 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|
+-----------+------+----------------+
only showing top 20 rows
```

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

```python
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 lamda fundtion

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()
```

7.5.4.6 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.

7.6 ADVANCED HADOOP

7.6.1 Amazon EMR (Elastic Map Reduce)〇 ☁

Amazon EMR facilitates you to analyze and process vast(huge) amounts of data by distributing the computational work across a cluster of virtual servers running in the AWS Cloud. The EMR cluster is managed using an open-source framework called Hadoop. Amazon EMR lets you focus on crunching or analyzing your data without having to worry about time-consuming setup, management, and tuning of Hadoop clusters or the compute capacity they rely on unlike other Hadoop distributors like Cloudera,Hortonworks etc.,

- Easy: To maintain on demand basis
- Fast: Auto shrinking of cluster and dynamically increase memory based on the need
- Cost-effective: Scala out and in anytime based on the business requirement

or models

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
- Security
- KMS

7.6.1.1 Why EMR?

The following ar reasons given by Amazone for using EMR

- Easy to Use: Launch cluster in a 5 to 10 minutes time as many cluster of nodes as you need
- Pay as you go: Pay an hourly rate (with AWS latest pricing model, customers can choose to pay in minutes)
- Flexible: Easily Add/ Remove capacity(Auto scale out and in anytime)
- Reliable: Spend less time for monitoring and can utilize in-built AWS tools which will reduce overhead
- Secure: Manage firewall (VPC both private and subnet)

7.6.1.2 Understanding Clusters and Nodes

The component of Amazon EMR is the cluster. A cluster is a collection of Amazon Elastic Compute Cloud (Amazon EC2) instances. Each instance in the cluster is called a node. Each node has a role within the cluster, referred to as the node type. Amazon EMR also installs different software components on each node type, giving each node a role in a distributed application like Apache Hadoop.

The node types in Amazon EMR are as follows:

- Master node: A node that manages the cluster by running software components to coordinate the distribution of data and tasks among other nodes for processing. The master node tracks the status of tasks and monitors the health of the cluster. Every cluster has a master node, and it is possible to create a single-node cluster with only the master node.

- Core node: A node with software components that run tasks and store data in the Hadoop Distributed File System (HDFS) on your cluster. Multi-node clusters have at least one core node.

- Task node: A node with software components that only runs tasks and does not store data in HDFS. Task nodes are optional.

The following diagram represents a cluster with one master node and four core nodes.
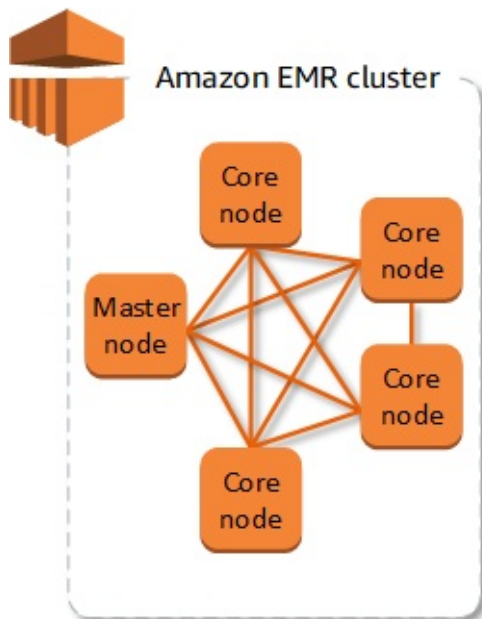
Cluser and Nodes

7.6.1.2.1 Submit Work to a Cluster

When you run a cluster on Amazon EMR, you have several options as to how you specify the work that needs to be done.

Provide the entire definition of the work to be done in functions that you specify

as steps when you create a cluster. This is typically done for clusters that process a set amount of data and then terminate when processing is complete.

Create a long-running cluster and use the Amazon EMR console, the Amazon EMR API, or the AWS CLI to submit steps, which may contain one or more jobs.

Create a cluster, connect to the master node and other nodes as required using SSH, and use the interfaces that the installed applications provide to perform tasks and submit queries, either scripted or interactively.

7.6.1.2.2 Processing Data

When you launch your cluster, you choose the frameworks and applications to install for your data processing needs. To process data in your Amazon EMR cluster, you can submit jobs or queries directly to installed applications, or you can run steps in the cluster.

- Submitting Jobs Directly to Applications:

  You can submit jobs and interact directly with the software that is installed in your Amazon EMR cluster. To do this, you typically connect to the master node over a secure connection and access the interfaces and tools that are available for the software that runs directly on your cluster. For more information, see Connect to the Cluster.

- Running Steps to Process Data

  You can submit one or more ordered steps to an Amazon EMR cluster. Each step is a unit of work that contains instructions to manipulate data for processing by software installed on the cluster.

The following is an example process using four steps:

1. Submit an input dataset for processing.
2. Process the output of the first step by using a Pig program.
3. Process a second input dataset by using a Hive program.
4. Write an output dataset.

Generally, when you process data in Amazon EMR, the input is data stored as files in your chosen underlying file system, such as Amazon S3 or HDFS. This data passes from one step to the next in the processing sequence. The final step writes the output data to a specified location, such as an Amazon S3 bucket.

Steps are run in the following sequence:

1. A request is submitted to begin processing steps.
2. The state of all steps is set to PENDING.
3. When the first step in the sequence starts, its state changes to RUNNING. The other steps remain in the PENDING state.
4. After the first step completes, its state changes to COMPLETED.
5. The next step in the sequence starts, and its state changes to RUNNING. When it completes, its state changes to COMPLETED.
6. This pattern repeats for each step until they all complete and processing ends.

The following diagram represents the step sequence and change of state for the steps as they are processed.
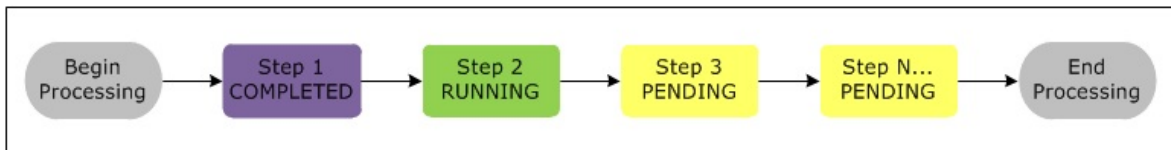
Cluser and Nodes

If a step fails during processing, its state changes to TERMINATED_WITH_ERRORS. You can determine what happens next for each step. By default, any remaining steps in the sequence are set to CANCELLED and do not run. You can also choose to ignore the failure and allow remaining steps to proceed, or to terminate the cluster immediately.

The following diagram represents the step sequence and default change of state when a step fails during processing.
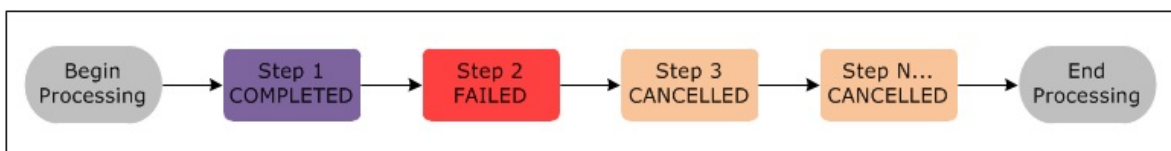
Cluser and Nodes

7.6.1.3 AWS Storage

S3 - Cloud based storage - Using EMRFS can directly connects s3 storage - Accessible from any where

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

7.6.1.4 Create EMR in AWS

7.6.1.4.1 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 50, Figure 51), click on S3 under Storage, Figure 52, Figure 53, Figure 54. Click on Create bucket button and then provide all the details to complete bucket creation.
- AWS Console

Figure 50: AWS Console

- AWS Login
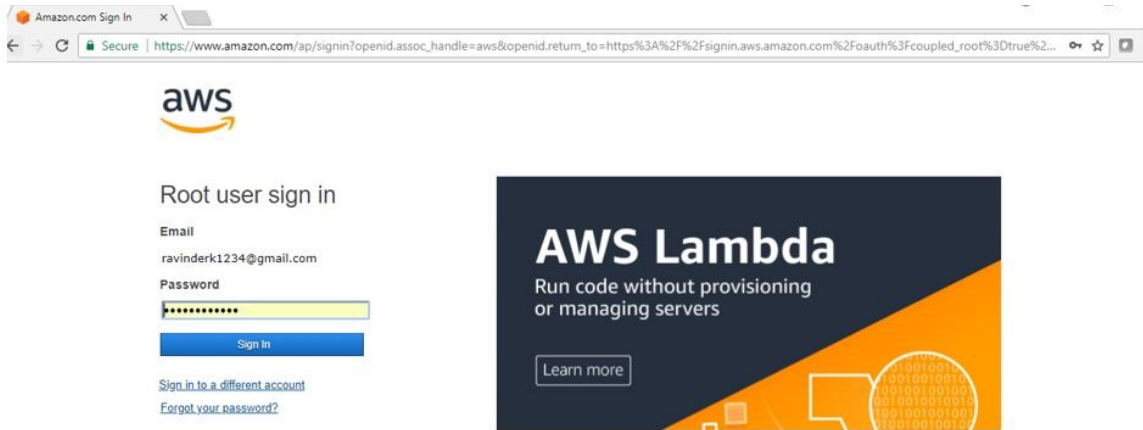
Figure 51: AWS Login

- S3 – Amazon Storage
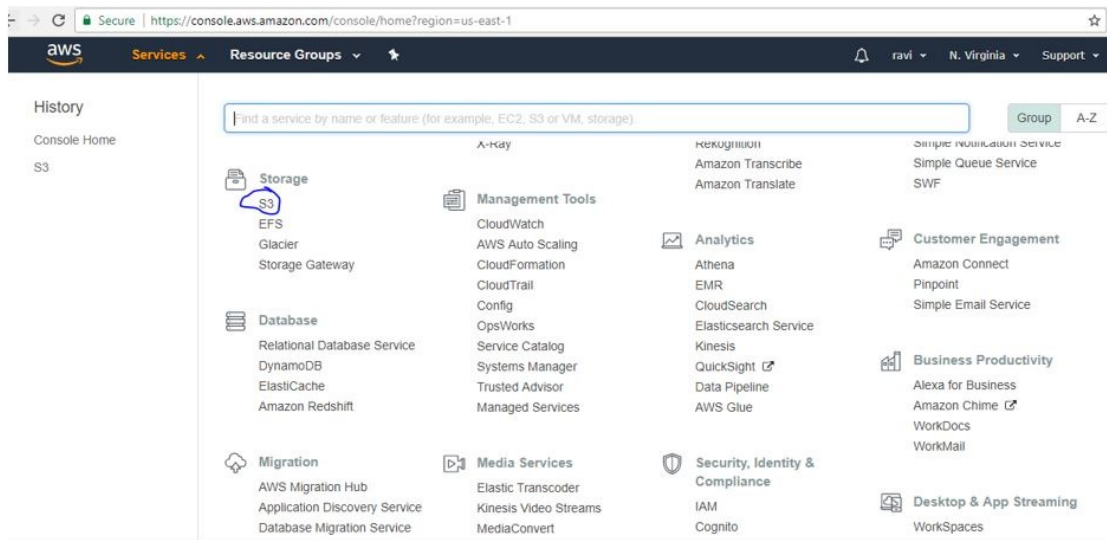
Figure 52: Amazon Storage

- S3 – Create buckets
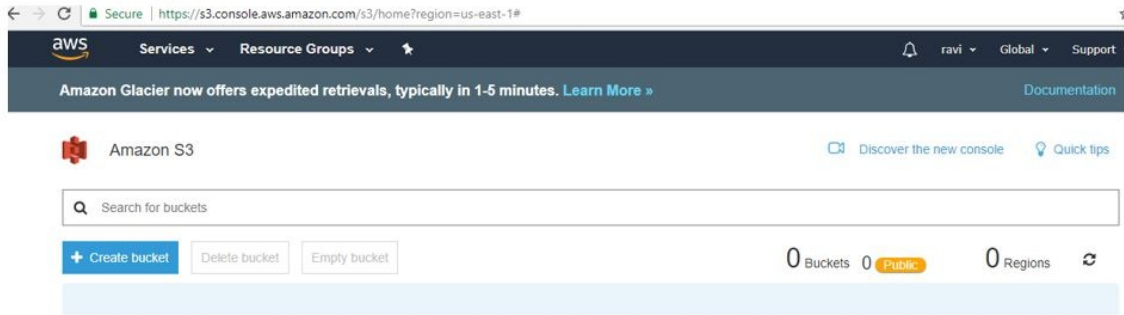
Figure 53: S3 buckets
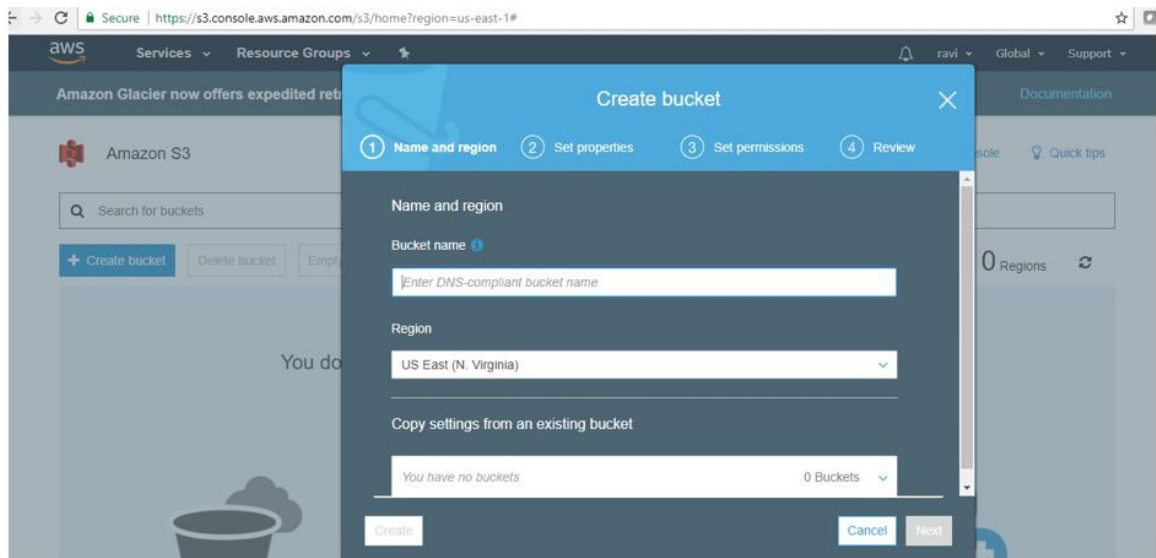
Figure 54: S3 buckets1

7.6.1.4.2 Create Key Pairs

- Login to AWS console, go to services, click on EC2 under compute. Select the Key pairs resoure, click on Create Key Pair and provide Key Pair name to complete the Key pairs creation. See Figure 55

- 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 56

7.6.1.4.2.1 Create Key Value Pair Screen shots
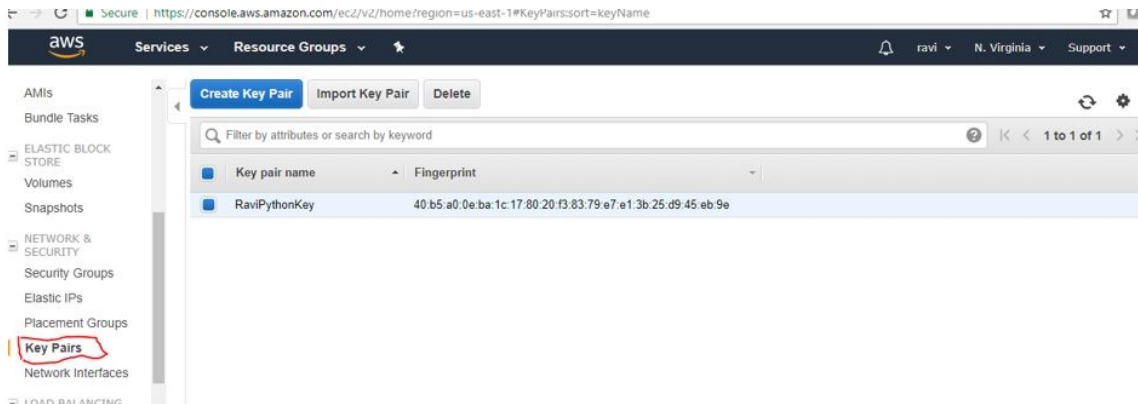
Figure 55: AMS Key Value Pair
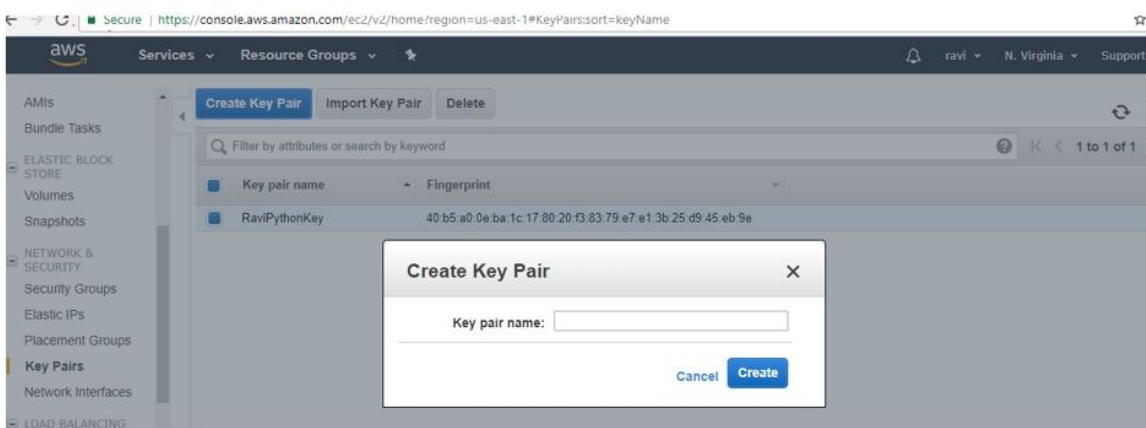
Figure 56: AMS Key Value Pair1

7.6.1.5 Create Step Execution – Hadoop Job

Login to AWS console, go to services and then select EMR. Click on Create Cluster. The cluster configuration provides details to complete to complete step execution creation. See: Figure 57, Figure 58, Figure 59, Figure 60, Figure 61

- Cluster name (Example: HadoopJobStepExecutionCluster)
- Select Logging check box and provide S3 folder location (Example: 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.
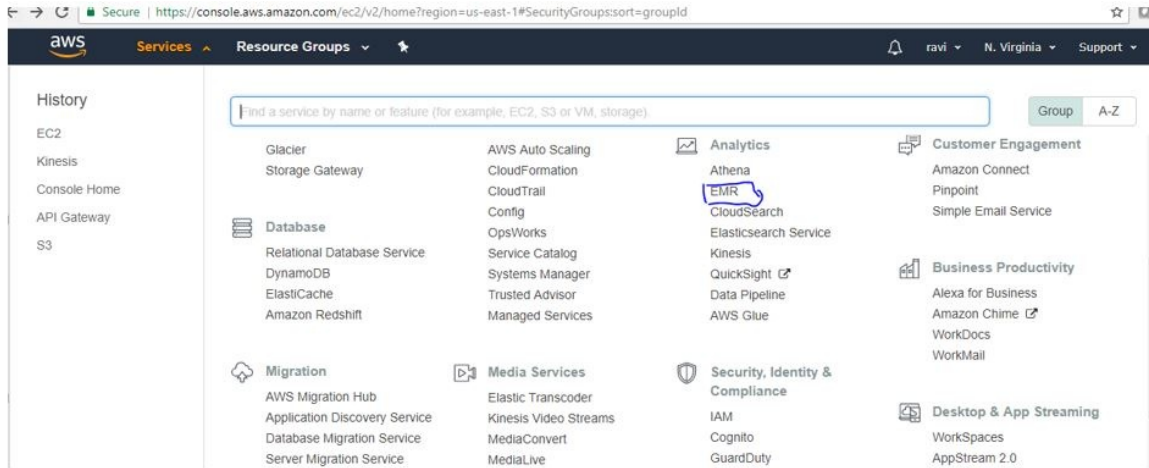
Figure 57: AWS EMR
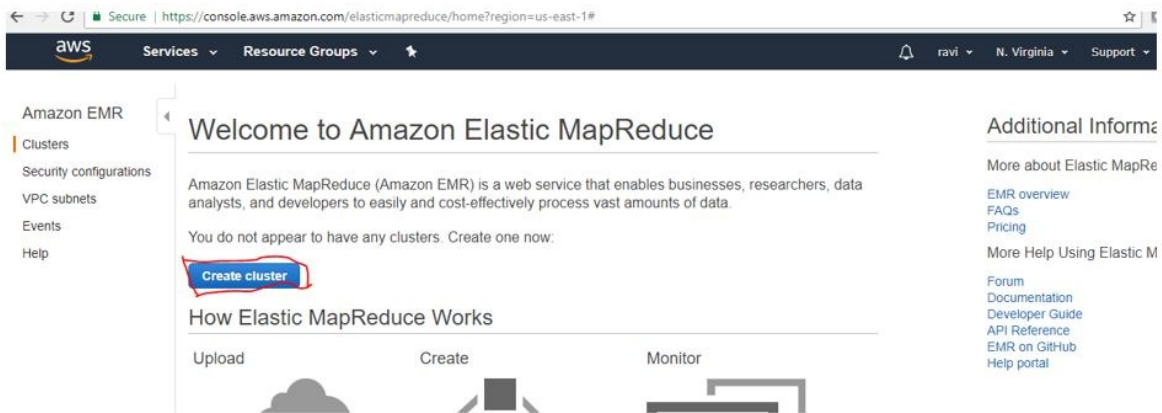
Figure 58: AWS Create EMR
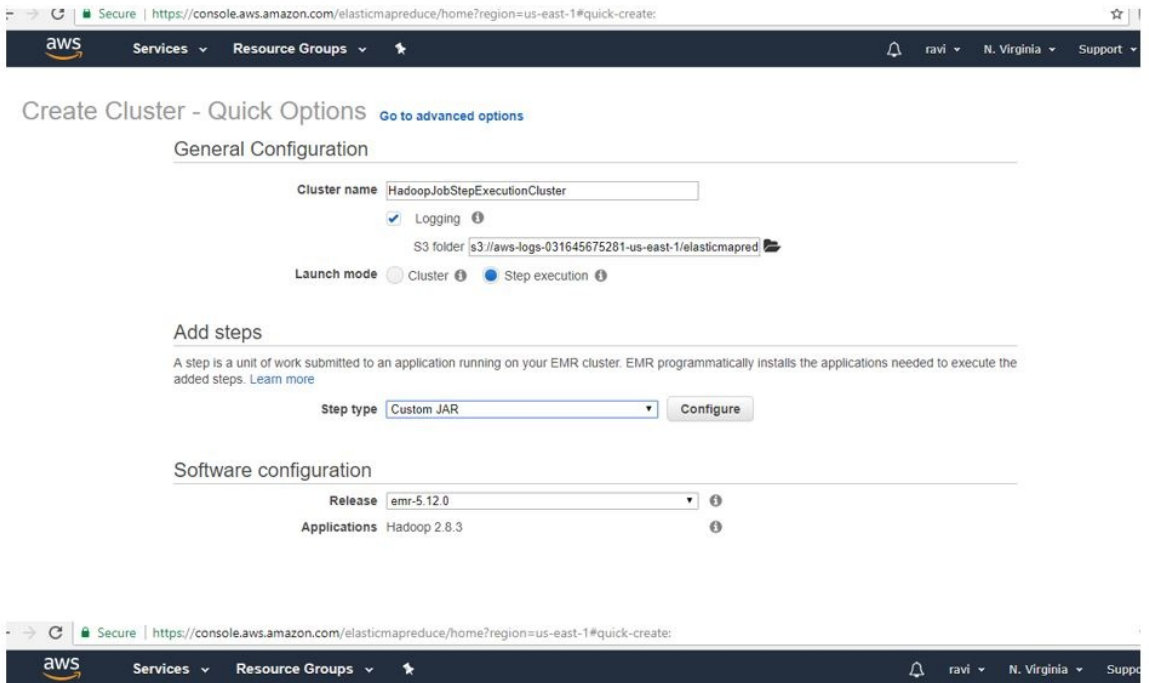
Figure 59: AWS Config EMR
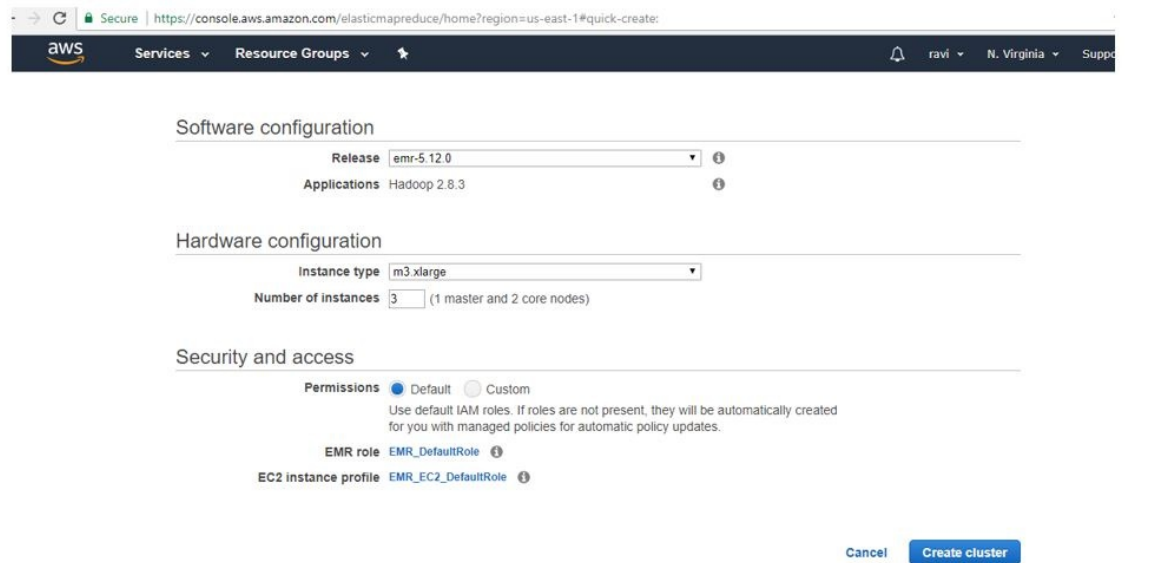
Figure 60: AWS Create Cluster
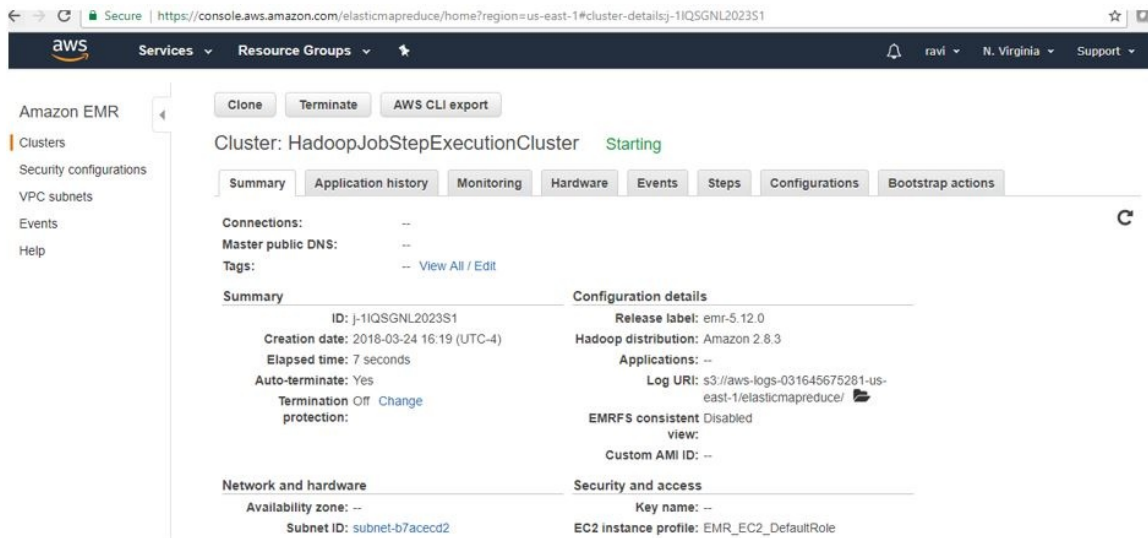
Figure 61: AWS Create Cluster1

7.6.1.6 Create a Hive Cluster

Login to AWS console, go to services and then select EMR. Click on Create Cluster. The cluster configuration provides details to complete. See, Figure 62, Figure 63, Figure 64

- Cluster name (Example: 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

7.6.1.6.1 Create a Hive Cluster - Screen shots
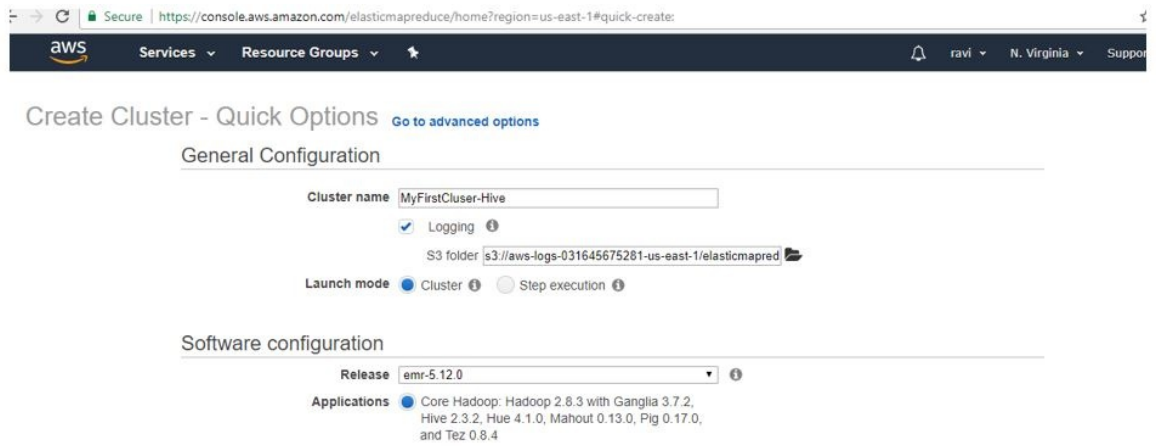
Figure 62: Hive Cluser
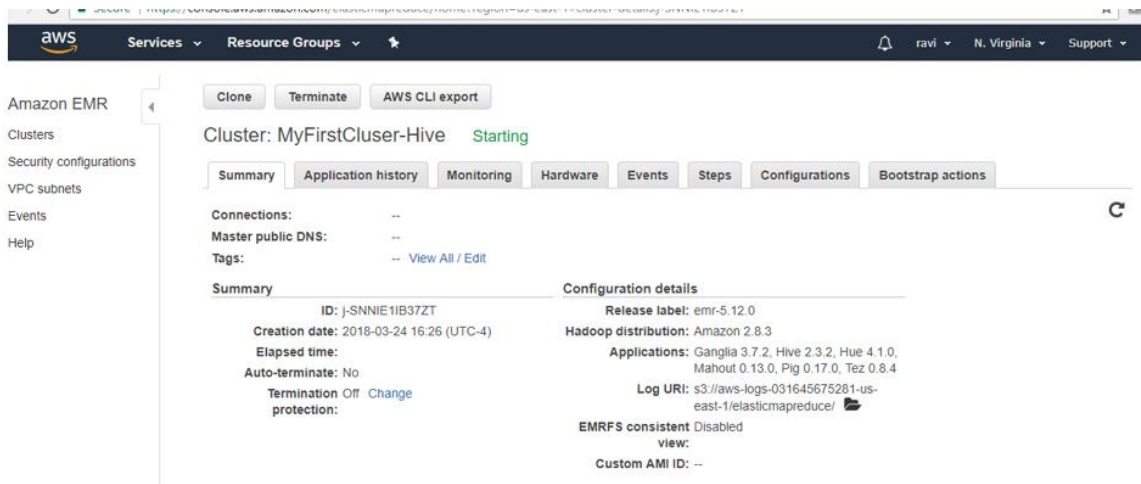
Figure 63: Hive Cluser1
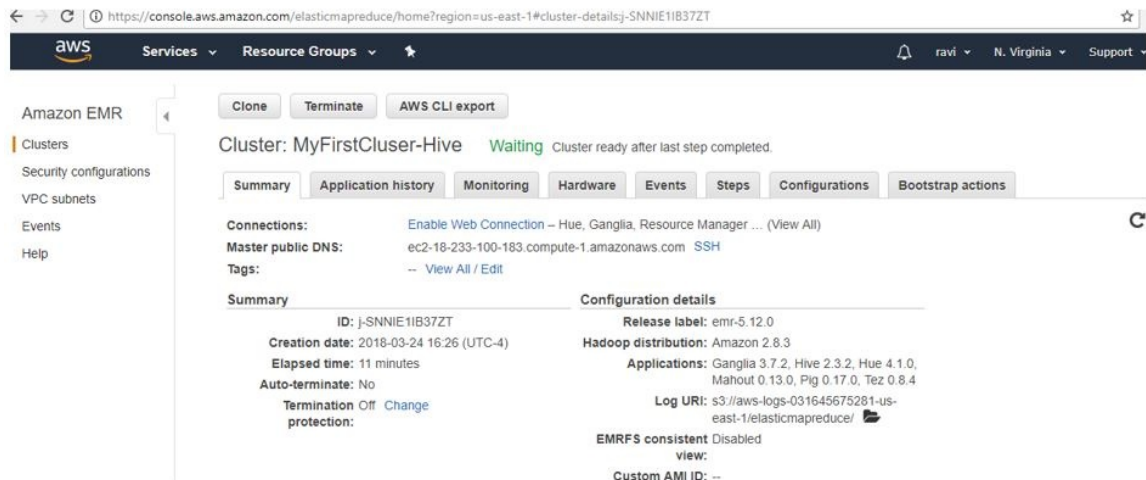
Figure 64: Hive Cluser2

7.6.1.7 Create a Spark Cluster

Login to AWS console, go to services and then select EMR. Click on Create Cluster. The cluster configuration provides details to complete. See, Figure 65, Figure 66, Figure 67

- Cluster name (Example: 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

7.6.1.7.1 Create a Spark Cluster - Screenshots
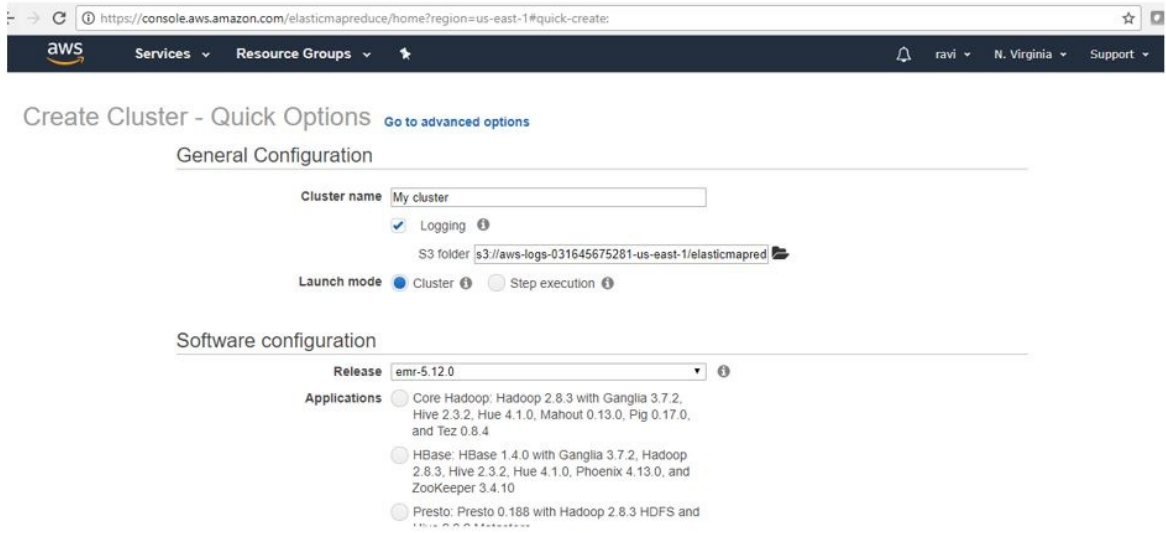
Figure 65: Spark Cluser
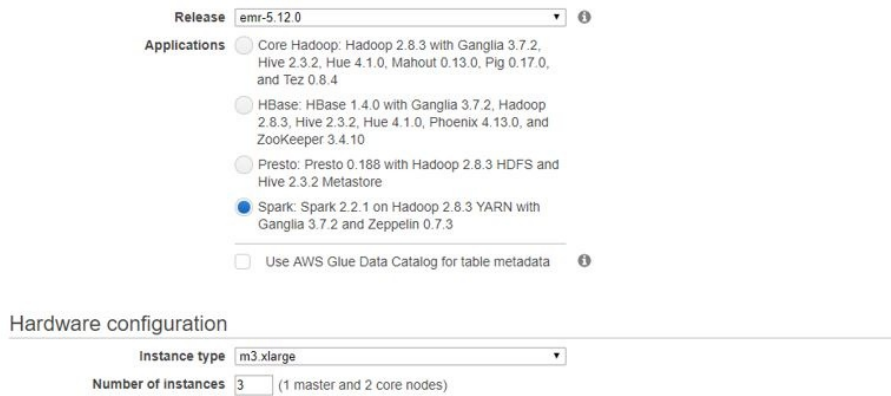
Figure 66: Spark Cluser
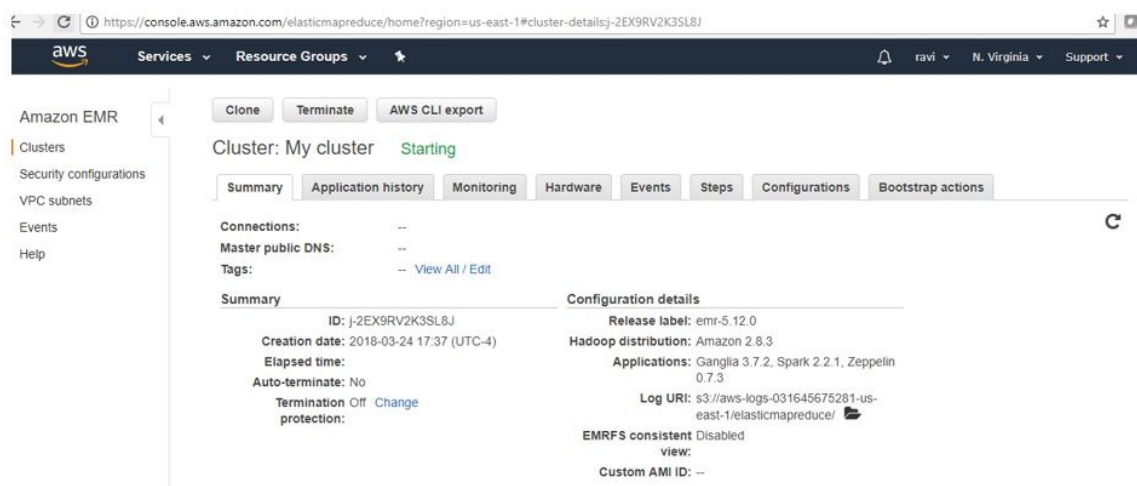
Figure 67: Spark Cluser

7.6.2 Twister2

7.6.2.1 Introduction

Twister2[57] provides a data analytics hosting environment where it supports different data analytics including streaming, data pipelines and iterative computations. The functionality of Twister2 is similar to other Big data frameworks such as Apache Spark and Apache Flink. But there are a few key differences which differentiates Twister2 from other frameworks. Unlike many other big data systems that are designed around user APIs, Twister2 is built from bottom up to support different APIs and workloads. The aim of Twister2 is to develop a complete computing environment for data analytics.

One major goal of Twister2 is to provide independent components, that can be used by other big data systems and evolve separately. To this end Twister2 supports a composable architecture where developers can easily replace a small component in the system with a new implementation very easily. For example the resource scheduling layer has several implementations it supports, Kubernetes, Mesos, Slurm, Nomad and a standalone implementation, If a user wants to add support for another resources scheduler such as Yarn they can easily do so by implementing the well defined interfaces.

Twister2 supports both batch and streaming applications. Unlike other big data frameworks which either support batch or streaming in the core and develop the

other on top of that, Twister2 natively supports both batch and streaming. Which allows Twister2 to make separate optimizations for each type.

Twister2 project is still less than 2 years old and still in it's early stages and going through rapid development to complete its functionality. It is an Open Source project which is licenced under the Apache 2.0[58]

7.6.2.2 Twister2 API's

Twister2 provides users with 3 levels on API's which can be used to write applications. The 3 API levels are shown in Figure Figure 68.
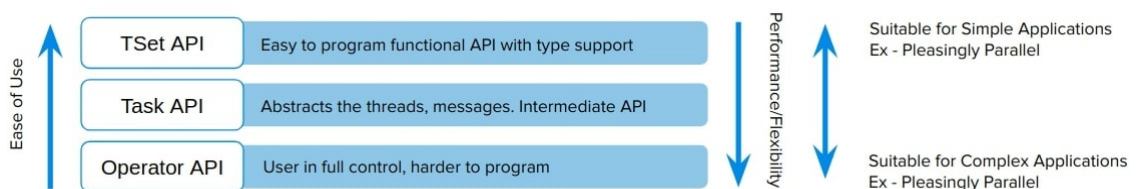
Figure 68: Twister2 API's

As shown in Figure 68 each API level has different levels of abstraction and programming complexities. TSet API is the most high level in Twister2 which in someways is simlar to the RDD API in Apache Spark or DataSet API in Apache Flink. If the user wants more control over the application development they can opt to use a more lower level API's.

7.6.2.2.1 TSet API

TSet API is the most abstract API provided by Twister2. This allows user to develop their programs at the data layer, similar to the programming model of Apache Spark. Similar to RDD in Spark users can perform operations on top of TSet objects which will be automatically parallelized by the framework. To get a slight understanding of the Tset API take a look at the abstract example given on how TSet API can be used to implement KMeans algorithm.

```java
public class KMeansJob extends TaskWorker {
    //......
    @Override
    public void execute() {
        //.....
        TSet<double[][]> points = TSetBuilder.newBuilder(config).createSource(new Source<double[][]>() {
            //Code for source function to read data points
        }).cache();
```

```
TSet<double[][]> centroids = TSetBuilder.newBuilder(config).createSource(new Source<double[][]>() {
    //Code for source function to read centers (or generate random centers)
}).cache();

for (
        int i = 0;
        i < iterations; i++) {
    TSet<double[][]> KmeansTSet = points.map(new MapFunction<double[][], double[][]>() {
        //Code for Kmeans calculation, this will have access to the centroids which are passed in
    });
    KmeansTSet.addInput("centroids", centroids);

    Link<double[][]> allReduced = KmeansTset.allReduce();
    TSet<double[][]> newCentroids = allReduced.map(new MapFunction<double[][], Object>() {
    /* Code that produces the new centers for the next iteration. The allReduce will result in
     a sum or all the centers sent by each worker so this map function simply needs to compute the
     average to get the new centers
     */
    });
    centroids.override(newCentroids);
}
//.....
    }
}
```

When programming at the TSet API level the developer does need to handle any information related to task and communications.

Note: The TSet API is currently under development and has not been released yet and therefore the API may change from what was discussed in this section, anyone who is interested can follow the development progress or contribute to the project through the GitHub repo[58].

7.6.2.2.2 Task API

The Task API allows developers to create their application at the Task level. The developer is responsible of managing task level details when developing at this API level, the upside of using the Task API is that it is more flexible than the TSet API so it allows developers to add custom optimizations to the application code. The TSet API is built on top of the Task API therefore the added layer of abstraction is bound to add slightly more overheads to the runtime, which you might be able to avoid by directly coding at the Task API level.

To get a better understanding of the Task API take a look at how the classic map reduce problem word count is implemented at using the Task API in the following code segment. This is only a portion of the example code, you can find the complete code for the example at[59].

```
public class WordCountJob extends TaskWorker {
//.....
  @Override
  public void execute() {
    // source and aggregator
```

```
    WordSource source = new WordSource();
    WordAggregator counter = new WordAggregator();

    // build the task graph
    TaskGraphBuilder builder = TaskGraphBuilder.newBuilder(config);
    builder.addSource("word-source", source, 4);
    builder.addSink("word-aggregator", counter, 4).keyedReduce("word-source", EDGE,
        new ReduceFn(Op.SUM, DataType.INTEGER), DataType.OBJECT, DataType.INTEGER);
    builder.setMode(OperationMode.BATCH);

    // execute the graph
    DataFlowTaskGraph graph = builder.build();
    ExecutionPlan plan = taskExecutor.plan(graph);
    taskExecutor.execute(graph, plan);
  }
  //.....
}
```

More Task API examples can be found in Twister2 documentations[60].

7.6.2.3 Operator API

The lowest level API provided by Twister2 is the Operator API, this allows developers to develop applications at the communication level. However since this API only abstracts out communication operations, details such as task management need to be handled by the application developer. Again similar to the Task API this provides the developer with more flexibility to create more optimized applications, at the cost of being harder to program. Twister2 supports a variety of communication patterns, known as collective communications in the HPC world. These communications are highly optimized using various routing patterns to reduce the number of communication calls that go through the network to provide users with a extremely efficient Operator API. The following list show the communication operations that are supported by Twister2. You can find more information on each or these operations in the Twister2 documentation[61].

- Reduce
- Gather
- AllReduce
- AllGather
- Partition
- Broadcast
- Keyed Reduce
- Keyed Partition
- Keyed Gather

Initial Performance comparisons that are discussed in[62] show how Twister2 out performs popular frameworks such Apache Flink, Apache Spark and Apache Strom in many areas. For example the Figure 69 shows a comparision between Twister2, MPI and Apache Spark versions of KMeans algorithm, please note that the graph is in logarithmic scale
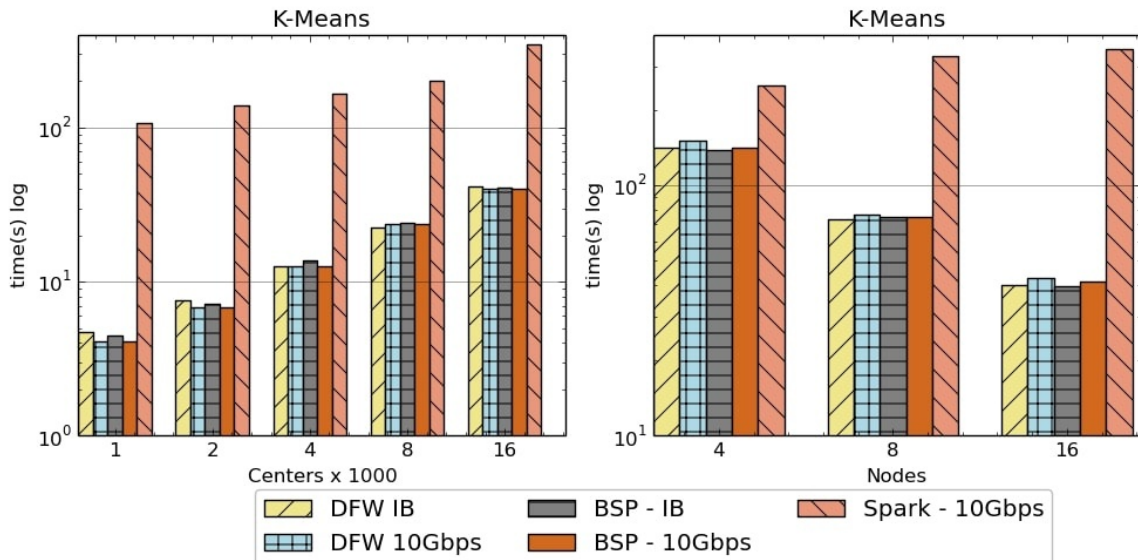
Figure 69: Kmeans Performance Comparison[63]

Notation : * **DFW** refers to Twister2 * **BSP** refers to MPI (OpenMPI)

This shows that Twister2 performs around ~10x faster than Apache Spark for KMeans. And that it is on par with implementations done using OpenMPI which is a widely used HPC framework.

7.6.2.3.1 Resources

- http://www.iterativemapreduce.org/

- http://www.cs.allegheny.edu/sites/amohan/teaching/CMPSC441/paper10.pd

- https://twister2.gitbook.io/twister2/

- http://dsc.soic.indiana.edu/publications/Twister2.pdf

- https://www.computer.org/csdl/proceedings/cloud/2018/7235/00/723501a38 abs.html

7.6.3 Twister2 Installation ☁

7.6.3.1 Prerequisites

Because Twister2 is still in the early stages of development a binary release is not available as of yet, therefore to try out Twister2 users need to first build the binaries from the source code.

- Operating System :
  - Twister2 is tested and known to work on,
  - Red Hat Enterprise Linux Server release 7
  - Ubuntu 14.05, Ubuntu 16.10 and Ubuntu 18.10

- 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](#)

7.6.3.1.1 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
```

7.6.3.1.2 OpenMPI Installation

When you compile Twister2 it will automatically download and compile OpenMPI 3.1.2 with it. If you don't like this version of OpenMPI and wants to use your own version, you can compile OpenMPI using following instructions.

- We recommend using OpenMPI 3.1.2
- Download OpenMPI 3.0.0 from https://download.open-mpi.org/release/open-mpi/v3.1/openmpi-3.1.2.tar.gz
- Extract the archive to a folder named openmpi-3.1.2
- Also create a directory named build in some location. We will use this to install OpenMPI
- Set the following environment variables

```
BUILD=<path-to-build-directory>
OMPI_312=<path-to-openmpi-3.1.2-directory>
PATH=$BUILD/bin:$PATH
LD_LIBRARY_PATH=$BUILD/lib:$LD_LIBRARY_PATH
export BUILD OMPI_312 PATH LD_LIBRARY_PATH
```

- The instructions to build OpenMPI depend on the platform. Therefore, we highly recommend looking into the `$OMPI_1101/INSTALL` file. Platform specific build files are available in `$OMPI_1101/contrib/platform` directory.

- In general, please specify `--prefix=$BUILD` and `--enable-mpi-java` as arguments to configure script. If Infiniband is available (highly recommended) specify `--with-verbs=<path-to-verbs-installation>`. Usually, the path to verbs installation is `/usr`. In summary, the following commands will build OpenMPI for a Linux system.

```
cd $OMPI_312
./configure --prefix=$BUILD --enable-mpi-java
make -j 8;make install
```

- If everything goes well `mpirun --version` will show `mpirun (Open MPI) 3.1.2`. Execute the following command to instal `$OMPI_312/ompi/mpi/java/java/mpi.jar` as a Maven artifact.

```
mvn install:install-file -DcreateChecksum=true -Dpackaging=jar -Dfile=$OMPI_312/ompi/mpi/java/java/mpi.jar -DgroupId=ompi
```

7.6.3.1.3 Install Extras

Install the other requirements as follows,

sudo apt-get install g++ git build-essential automake cmake libtool-bin zip libunwind-setjmp0-dev zlib1g-dev unzip pkg-config python-setuptools -y sudo apt-get install python-dev python-pip

Now you have successfully installed the required packages. Let us compile Twister2.

7.6.3.1.4 Compiling Twister2

Now lets get a clone of the source code.

```
git clone https://github.com/DSC-SPIDAL/twister2.git
```

You can compile the Twister2 distribution by using the bazel target as follows.

```
cd twister2
bazel build --config=ubuntu scripts/package:tarpkgs
```

This will build twister2 distribution in the file

```
bazel-bin/scripts/package/twister2-client-0.1.0.tar.gz
```

If you would like to compile the twister2 without building the distribution packages use the command

```
bazel build --config=ubuntu twister2/...
```

For compiling a specific target such as communications

```
bazel build --config=ubuntu twister2/comms/src/java:comms-java
```

7.6.3.1.5 Twister2 Distribution

After you've build the Twister2 distribution, you can extract it and use it to submit jobs.

```
cd bazel-bin/scripts/package/
tar -xvf twister2-0.1.0.tar.gz
```

7.6.4 Twister2 Examples ☁

Twister documentation lists several examples[64] that users can leverage to better understand the Twister2 API's. Currently there are several

Communication API examples and Task API examples available in the Twister2 documentation. In this section we will go through how an example can be executed with Twister2.

7.6.4.1 Submitting a Job

In order to run an example users need to submit the example to Twister2 using the `twister` command. This command is found inside the bin directory of the distribution.

Here is a description of the command

```
twister2 submit cluster job-type job-file-name job-class-name [job-args]
```

- submit is the command to execute
- cluster which resource manager to use, i.e. standalone, kubernetes, this should be the name of the configuration directory for that particular resource manager
- job-type at the moment we only support jar
- job-file-name the file path of the job file (the jar file)
- job-class-name name of the job class with a main method to execute

Here is an example command

```
./bin/twister2 submit standalone jar examples/libexamples-java.jar edu.iu.dsc.tws.examples.task.ExampleTaskMain -itr 80 -
```

In this command, cluster is standalone and has program arguments.

For this exercise we are using the standlone mode to submit a job. However Twister2 does support Kubernetes, Mesos, Slurm and Nomad resource schedulers if users want to submit jobs to larger cluster deployments.

7.6.4.2 Batch WordCount Example

In this section we will run a batch word count example from Twister2. This example only uses communication layer and resource scheduling layer. The threads are managed by the user program.

The example code can be found in

```
twister2/examples/src/java/edu/iu/dsc/tws/examples/basic/batch/wordcount/
```

When we install Twister2, it will compile the examples. Lets go to the installation directory and run the example.

```
cd bazel-bin/scripts/package/twister2-dist/
./bin/twister2 submit standalone jar examples/libexamples-java.jar edu.iu.dsc.tws.examples.batch.wordcount.WordCountJob
```

This will run 4 executors with 8 tasks. So each executor will have two tasks. At the first phase, the 0-3 tasks running in each executor will generate words and after they are finished, 5-8 tasks will consume those words and create a count.

7.6.5 HADOOP RDMA ☁

Acknowledgement: This section was copied and modified with permission from https://www.chameleoncloud.org/appliances/17/docs/

In Chameleon cloud it is possible to launch a virtual Hadoop cluster on bare-metal InfiniBand nodes with SR-IOV.

The CentOS 7 SR-IOV RDMA-Hadoop is based on 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

We refer to the chameleon cloud bare metal user guide for documentation on how to reserve and provision resources using the appliance of CC-CentOS7-RDMA-Hadoop.

🅾 link missing

7.6.5.1 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-cores-per-VM> <ips-file> <openst
```

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.

7.6.5.2 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 next:

```
[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/ConnectX-3 Pro Virtual Function]
...
```

The PCI device ID of the Virtual Function is 03:00:1 in the previous 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 previous 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.

7.6.5.3 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.

7.6.5.4 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 next. 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{vernatim}

If you launched VMs using the start-vm.sh script, you should first manually kill all the VMs. Then, delete all the create

[user@host]$ neutron port-delete PORT
```

Please note that it is important to delete unused ports after experiments.

8 CONTAINER

8.1 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.

8.1.1 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](#)

8.1.2 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](#)

8.1.3 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](#)

8.1.4 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](#)

8.2 DOCKER

8.2.1 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/
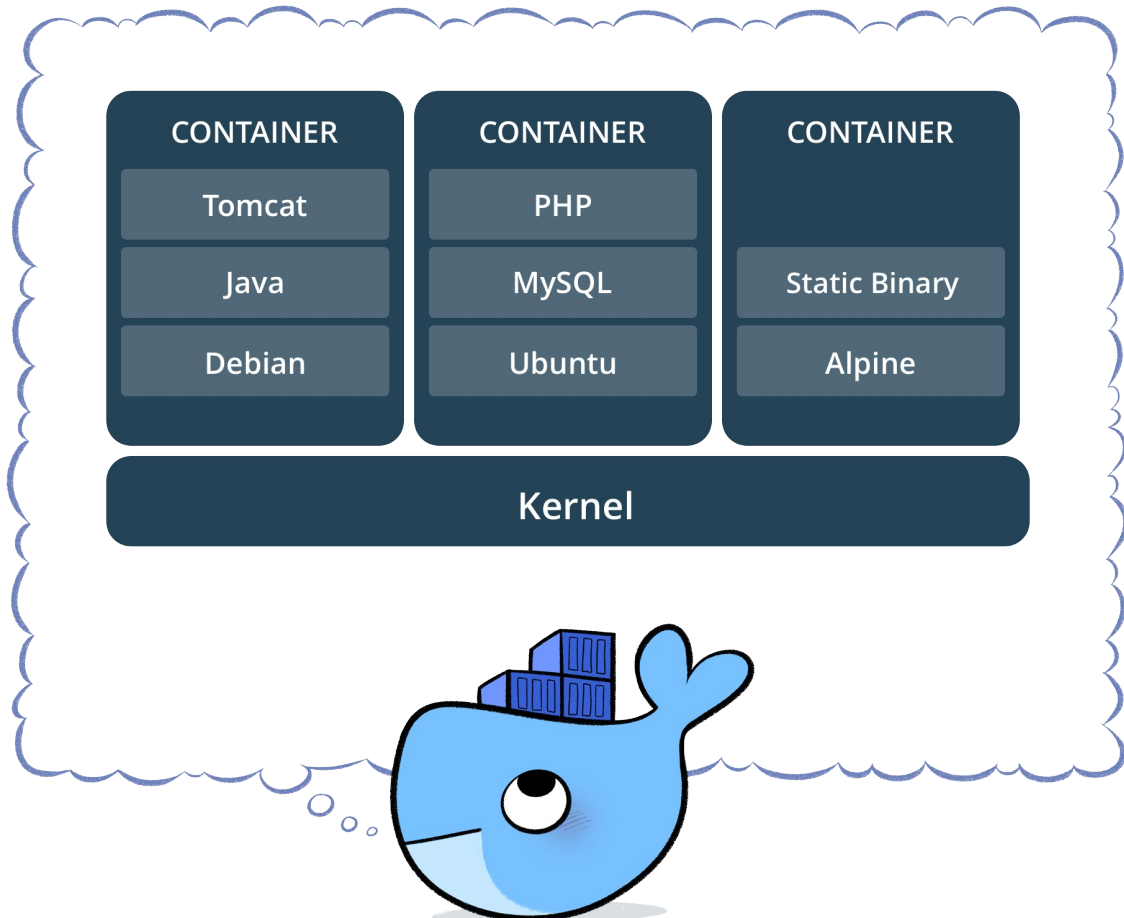
Figure 70: Docker Containers [Image Source] [65]

[Figure 70](#) 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.

8.2.1.1 Docker Engine

The Docker engine can be thought of as the core of the docker runtime. The docker engine mainly provides 3 services. [Figure 71](#) shows how the docker engine is composed.

- A long running server which manages the containers
- A REST API
- A command line interface
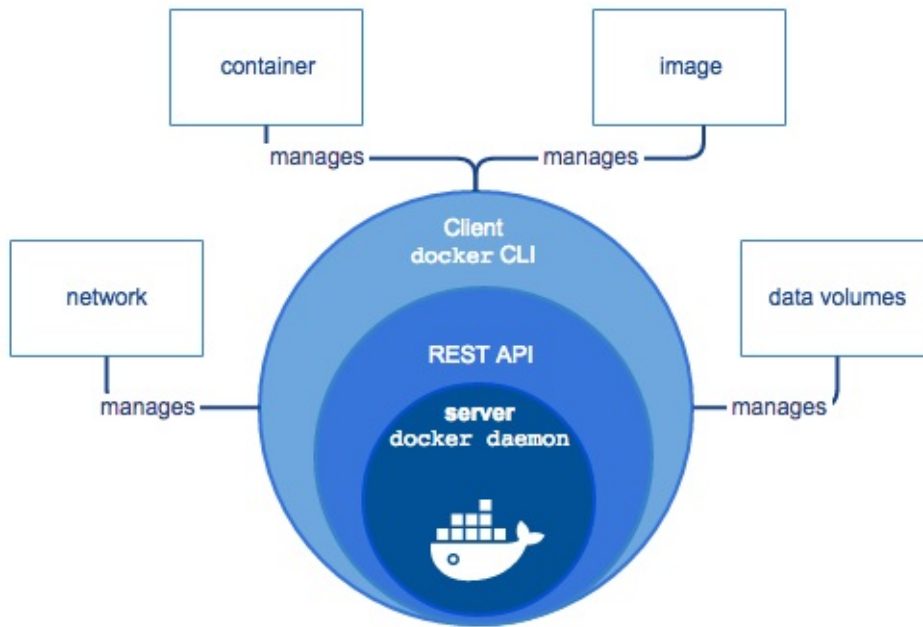
Figure 71: Docker Engine Component Flow [Image Source] [65]

8.2.1.2 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, bulding containers, etc. are responsibilities of the Docker daemon. Figure 72 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.
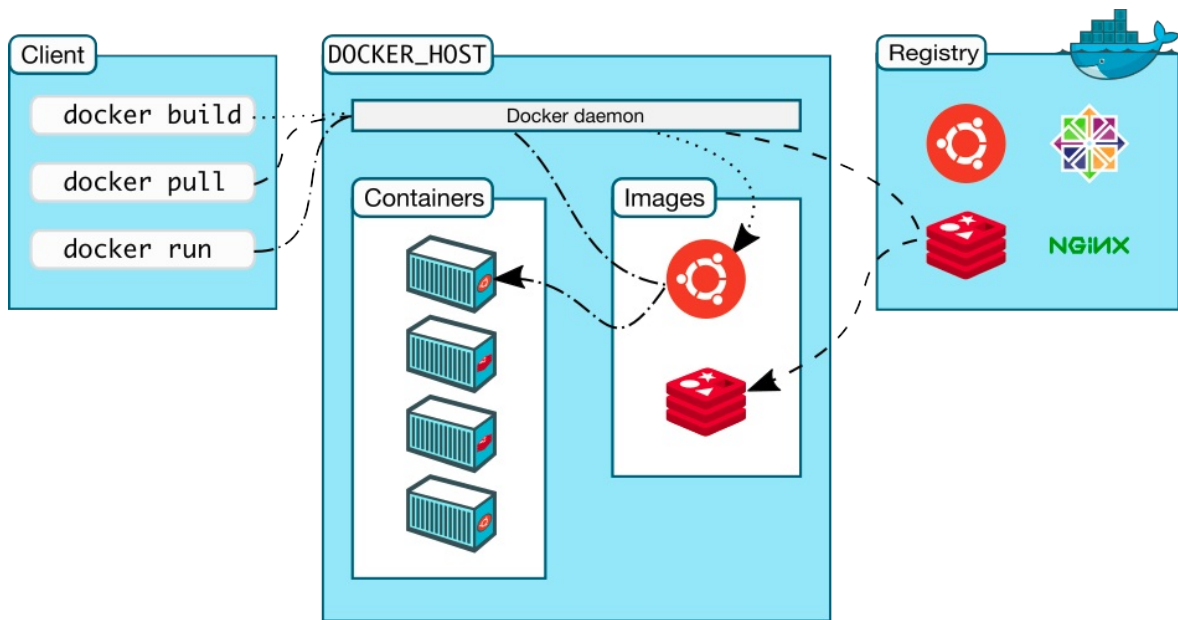
Figure 72: Docker Architecture [Image Source] [65]

8.2.1.3 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 73.

- https://www.docker.com/survey-2016
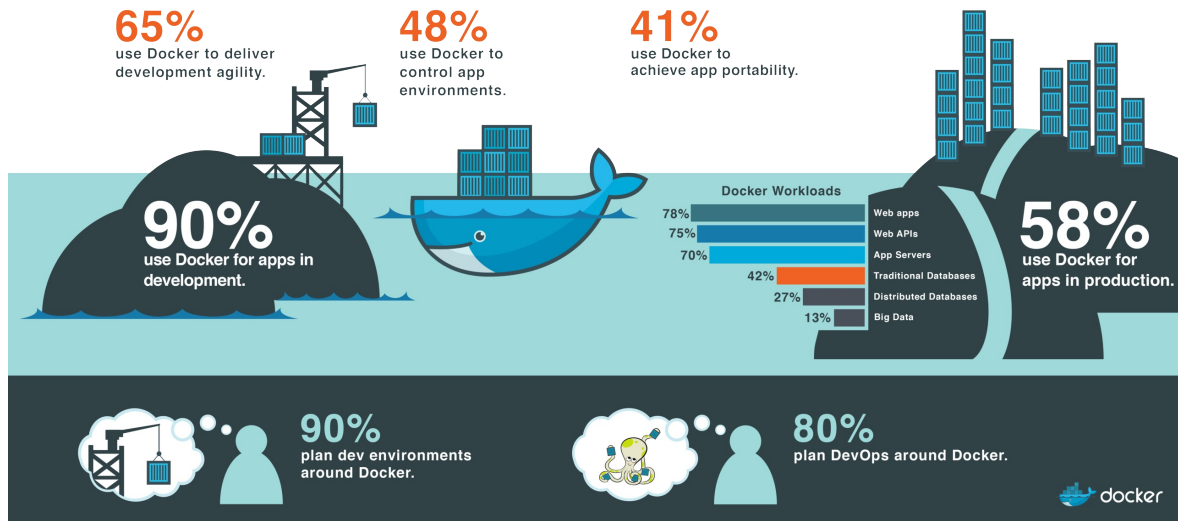
Figure 73: Docker Survey Results 2016 [Image Source] [65]

8.2.2 Running Docker Locally ☁

⚠ Please verify if the instructions are still up to date. Rapid changes could mean they can be outdated quickly. Also we assume the ubuntu instalations may have changed and may be different between 18.04 and 19.04.

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).

8.2.2.1 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

8.2.2.2 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 the next.

```
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
```

8.2.2.3 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%20In
- Follow the Wizard steps in the installer
- Launch docker
- Docker usually lauches automatically during windows startup.

8.2.2.4 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/
```

8.2.3 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

8.2.3.1 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

```python
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 sop all instances using it and the 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 that 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
```

8.2.3.2 References

The reference documentation about docker files can be found at

- https://docs.docker.com/engine/reference/builder/

8.2.4 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" [65]. 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 [66]. It allows users to download (pull), build, test and store their images for easy deployment on any host they may have [65].

8.2.4.1 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 distriuute 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 easiset if you set it in your shell with

```
local$ export DOCKERHUB=<PUT YOUR DOCKER USERNAME HERE>
```

8.2.4.2 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 previous 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.

8.2.4.3 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

8.2.4.4 Create Repositories

In order to push images to Docker Hub, you need to have a and 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 [67].

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

8.2.4.5 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	1.79GB
$DOCKERHUB/cloudmesh	v1.0	1f26a5f7a1b4	10 d ago	1.79GB
centos	latest	26cb1244b171	2 w ago	195MB
centos	latest	2d194b392dd1	2 w ago	195MB

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]
18f9479cfc2c: 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

8.2.4.6 Resources

- The offical [Overview of Docker Hub](#) [65]
- Information about using docker repositories can be found at [Repositories on Docker Hub](#) [67]
- [How to Use DockerHub](#) [66]
- [Docker Tutorial Series](#) [68]

8.3 DOCKER AS PAAS

8.3.1 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*.

8.3.1.1 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.

8.3.1.2 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 indicatet 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 described in next 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.

8.3.1.3 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
```

8.3.1.4 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-1-5c3anju1pwx94054r3vx0v7j4obyuggfu2cmesnx
    192.168.99.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

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 previously 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      ERRORS
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
```

8.3.1.5 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-compse.yml`. Since we are in the vm we need to create the file using the terminal. follow the steps given next 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 pres 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

## 8.3.2 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.

### 8.3.2.1 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 serer, 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.

8.3.2.2 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_PASSWROD")
'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
```

8.3.2.3 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.

8.3.2.4 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.

8.3.2.5 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)*

8.3.3 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 3.1.1 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/3.1.1/Dockerfile

8.3.3.1 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/3.1.1
$ docker build -t cloudmesh/hadoop:3.1.1 .
```

The complete docker image for Hadoop consumes 1.5GB.

```
$ docker images


REPOSITORY       TAG   IMAGE ID     CREATED     SIZE
cloudmesh/hadoop 3.1.1 ba2c51f94348 1 hour ago 1.52GB
```

To use the image interactively you can start the container as follows:

```
$ docker run -it cloudmesh/hadoop:3.1.1 /etc/bootstrap.sh -bash
```

It may take a few minutes at first to download image.

8.3.3.2 Hadoop Configuration Files

The configuration files are included in the `conf` folder

8.3.3.3 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

8.3.3.4 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
```

8.3.3.5 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/3.1.1/examples

We explain the examples next

8.3.3.5.1 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, maximim, average, and standard derivation from values stored in a number of input files. Figure Figure 74 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 74: MapReduce example in Docker

8.3.3.5.1.1 Base Location

The example is available within the container at:

```
container$ cd /cloudmesh/examples/statistics
```

8.3.3.5.1.2 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.

8.3.3.5.1.3 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/examples/statistics/src/MinMaxAvgS
```

These commands simply prepare compiling the example code and the compiled class files are generated at the *dest* location.

8.3.3.5.1.4 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/ .
```

8.3.3.5.1.5 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
```

8.3.3.5.1.6 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_0002
18/02/28 23:48:51 INFO impl.YarnClientImpl: Submitted application application_1519877569596_0002
18/02/28 23:48:51 INFO mapreduce.Job: The url to track the job: http://f5e82d68ba4a:8088/proxy/application_1519877569596_
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.

8.3.3.5.1.7 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
```

8.3.3.5.1.8 Run Program with Multiple Input Files

The first run was done pretty quickly (1440 milliseconds took according to the previous sample result) 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 next:

```
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_0001
18/02/28 23:17:19 INFO impl.YarnClientImpl: Submitted application application_1519877569596_0001
18/02/28 23:17:19 INFO mapreduce.Job: The url to track the job: http://f5e82d68ba4a:8088/proxy/application_1519877569596_0001/
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
```

8.3.3.5.1.9 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
```

8.3.3.5.2 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.

8.3.3.6 Refernces

- The details of the new version is available from the official site at http://hadoop.apache.org/docs/r3.1.1/index.html

8.3.4 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
```

8.3.4.1 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
```

8.3.4.2 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.cgl.hadoop.pagerank.HadoopPageRank input.pagerank output.pagerank 5000 1
```

Result

```
$ hadoop fs -cat output.pagerank/part-r-00000
```

8.3.5 Apache Spark with Docker 🌐

8.3.5.1 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
```

8.3.5.2 Running the Image

In this step, we will launch a Spark container.

8.3.5.2.1 Running interactively

```
$ docker run -it -p 8088:8088 -p 8042:8042 -h sandbox sequenceiq/spark:1.6.0 bash
```

8.3.5.2.2 Running in the background

```
$ docker run -d -h sandbox sequenceiq/spark:1.6.0 -d
```

8.3.5.3 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

8.3.5.3.1 Run Spark in Yarn-Client Mode

```
$ spark-shell --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1
```

8.3.5.3.2 Run Spark in Yarn-Cluster Mode

```
$ spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client --driver-memory 1g --executor-memory 1g --e
```

8.3.5.4 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
```

8.3.5.5 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.

8.3.5.5.1 Launch Spark Interactive Shell

```
$ spark-shell --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1
```

8.3.5.5.2 Program in Scala

```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()
```

8.3.5.5.3 Launch PySpark Interactive Shell

```
$ pyspark --master yarn-client --driver-memory 1g --executor-memory 1g --executor-cores 1
```

8.3.5.5.4 Program in Python

```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()
```

8.3.5.6 Docker Spark Examples

8.3.5.6.1 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 us 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(0.15000000000000002, 0.15000000000000002, 0.15000000000000002)
DenseVector(9.2, 9.2, 9.2)
DenseVector(0.0, 0.0, 0.0)
DenseVector(9.05, 9.05, 9.05)
```

8.3.5.6.2 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.TezJobExecutionContext --conf up
```

8.3.5.6.3 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.tez.TezJobExecutionContext --co
```

8.3.5.7 Interactive Examples

Here we need a new image to work on. Let us 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
99eee56fda2f: 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=f22492fe7ab8206ac2223359e0603a0dff54d98096ab7930
[I 15:47:53.177 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[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=f22492fe7ab8206ac2223359e0603a0dff54d98096ab7930
```

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)
```

8.3.5.7.1 Stop Docker Container

```
$ docker stop spark
```

8.3.5.7.2 Start Docker Container Again

```
$ docker start spark
```

8.3.5.7.3 Remove Docker Container

```
$ docker rm spark
```

8.4 KUBERNETES

8.4.1 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/

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

8.4.1.1 What are containers?

The old way: Applications on host

The new way: Deploy containers
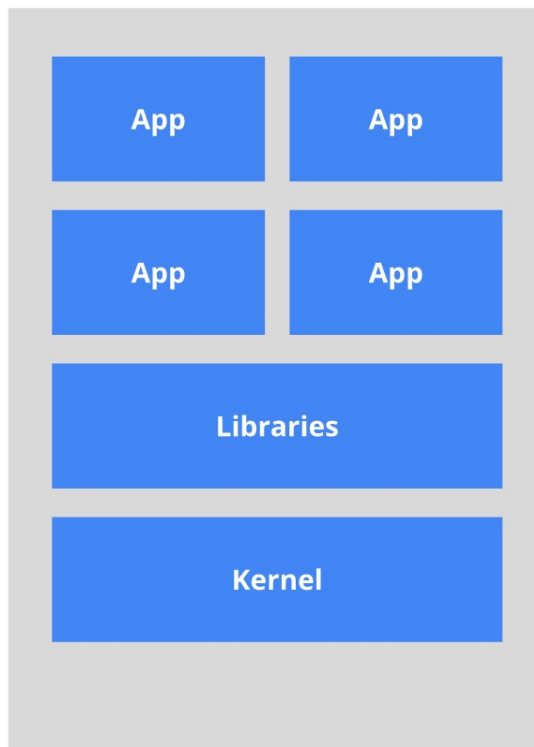
Heavyweight, non-portable
Relies on OS package manager
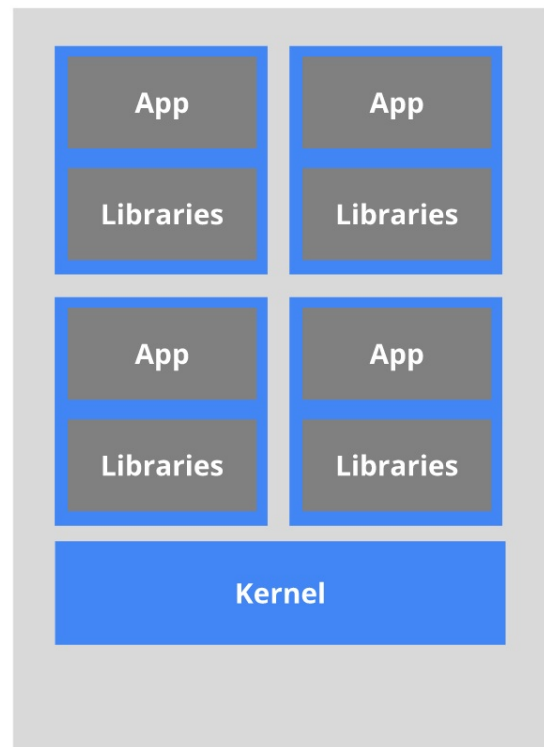
Small and fast, portable
Uses OS-level virtualization

Figure 75: Kubernetes Containers [Image Source]

Figure 75 shows a depiction of the container architecture.

8.4.1.2 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.

8.4.1.3 Kubernetes Architecture

The architecture of kubernets is shown in Figure 76.
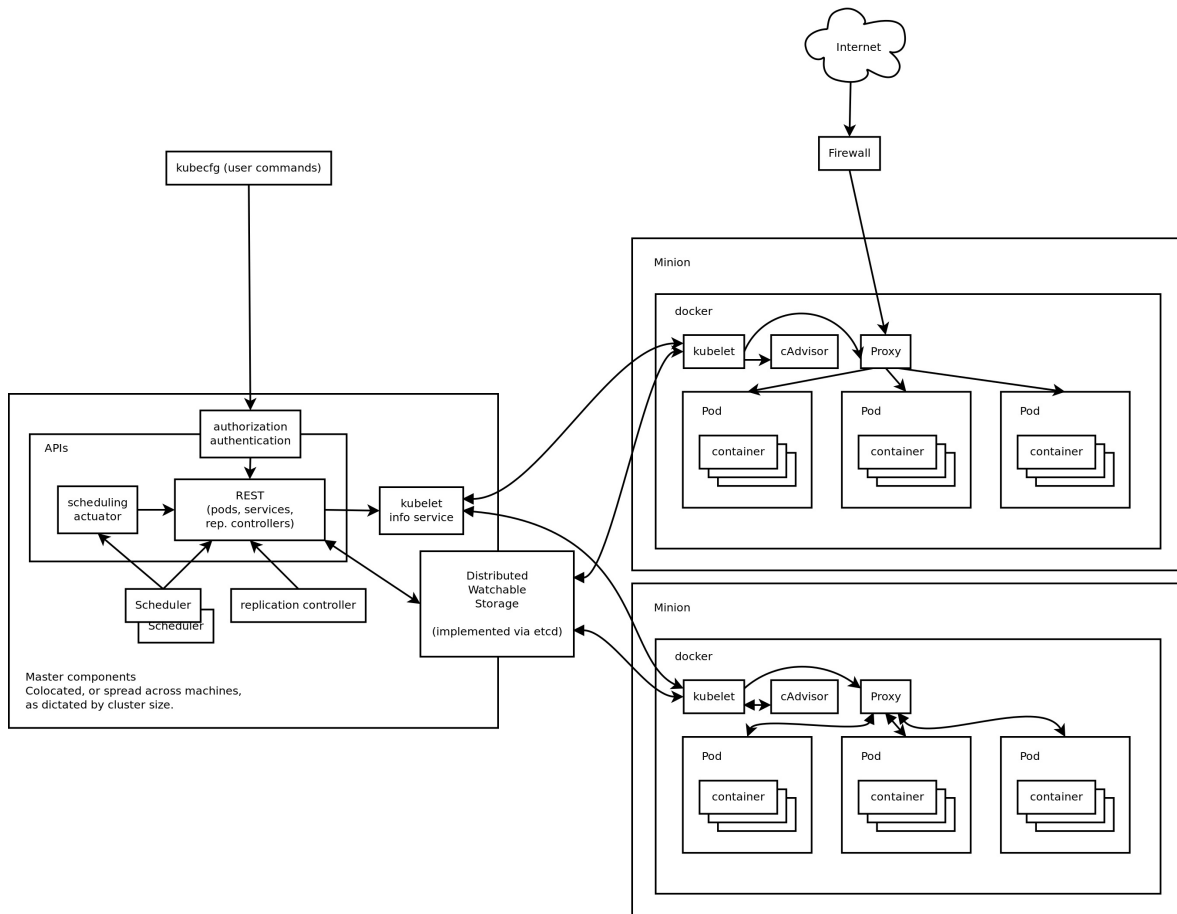
Figure 76: Kubernetes (Source: Google)

8.4.1.4 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.

8.4.1.4.1 Install minikube

8.4.1.4.1.0.1 OSX

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/minikube-darwin-amd64 && chmod +x minikube &
```

8.4.1.4.1.0.2 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.

8.4.1.4.1.0.3 Linux

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.25.0/minikube-linux-amd64 && chmod +x minikube &&
```

Installing KVM2 is important for Ubuntu distributions

```
$ sudo apt install libvirt-bin qemu-kvm
$ sudo usermod -a -G libvirtd $(whoami)
$ newgrp libvirtd
```

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-kvm2 && chmod +x docker-machine-
```

8.4.1.4.2 Start a cluster using Minikube

8.4.1.4.2.0.1 OSX Minikube Start

```
$ minikube start
```

8.4.1.4.2.0.2 Ubuntu Minikube Start

```
$ minikube start --vm-driver=kvm2
```

8.4.1.4.2.0.3 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"
```

8.4.1.4.3 Create a deployment

```
$ kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --port=8080
```

8.4.1.4.4 Expose the servi

```
$ kubectl expose deployment hello-minikube --type=NodePort
```

8.4.1.4.5 Check running status

This step is to make sure you have a pod up and running.

```
$ kubectl get pod
```

8.4.1.4.6 Call service api

```
$ curl $(minikube service hello-minikube --url)
```

8.4.1.4.7 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.

8.4.1.4.8 Delete the service and deployment

```
$ kubectl delete service hello-minikube
$ kubectl delete deployment hello-minikube
```

For all platforms we can use the following command.

```
$ minikube stop
```

8.4.1.5 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/

8.4.2 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*.

8.4.2.1 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 shown in next 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.

8.4.2.2 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 and 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 cn 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 se 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>
```

8.4.2.3 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.*

8.5 SINGULARITY

8.5.1 Running Singularity Containers on Comet ☁

This section was copied from

- https://www.sdsc.edu/support/user_guides/tutorials/singularity.html

and modified. To use it you will need an account on comet which can be obtained via XSEDE. In case you use this material as part of a class please contact your teacher for more information.

8.5.1.1 Background

What is Singularity?

> *"Singularity enables users to have full control of their environment. Singularity containers can be used to package entire scientific workflows, software and libraries, and even data. This means that you don't have to ask your cluster admin to install anything for you - you can put it in a Singularity container and run."*

[from the Singularity web site at http://singularity.lbl.gov/]

There are numerous good tutorials on how to install and run Singularity on Linux, OS X, or Windows so we won't go into much detail on that process here.

In this tutorial you will learn how to run Singularity on Comet. First we will review how to access a compute node on Comet and provide a simple example to help get you started. There are numerous tutorial on how to get started with Singularity, but there are some details specific to running Singularity on Comet which are not covered in those tutorials. This tutorial assumes you already have an account on Comet. You will also need access to a basic set of example files to get started. SDSC hosts a Github repository containing a 'Hello world!" example which you may clone with the following command:

```
git clone https://github.com/hpcdevops/singularity-hello-world.git
```

8.5.1.2 Tutorial Contents

- Why Singularity?
- Downloading & Installing Singularity
- Building Singularity Containers
- Running Singularity Containers on Comet
- Running Tensorflow on Comet Using Singularity

8.5.1.3 Why Singularity?

Listed next is a typical list of commands you would need to issue in order to implement a functional Python installation for scientific research:

```
COMMAND=apt-get -y install libx11-dev
COMMAND=apt-get install build-essential python-libdev
COMMAND=apt-get install build-essentyial openmpi-dev
COMMAND=apt-get install cmake
COMMAND=apt-get install g++
COMMAND=apt-get install git-lfs
COMMAND=apt-get install libXss.so.1
COMMAND=apt-get install libgdal1-dev libproj-dev
COMMAND=apt-get install libjsoncpp-dev libjsoncpp0
COMMAND=apt-get install libmpich-dev --user
COMMAND=apt-get install libpthread-stubs0 libpthread-stubs0-dev libx11-dev libx11-d
COMMAND=apt-get install libudev0:i386
COMMAND=apt-get install numpy
COMMAND=apt-get install python-matplotlib
COMMAND=apt-get install python3`
```

Singularity allows you to avoid this time-consuming series of steps by packaging these commands in a re-usable and editable script, allowing you to quickly, easily, and repeatedly implement a custom container designed specifically for your analytical needs.

Figure 77 compares a VM vs. Docker vs. Singularity.
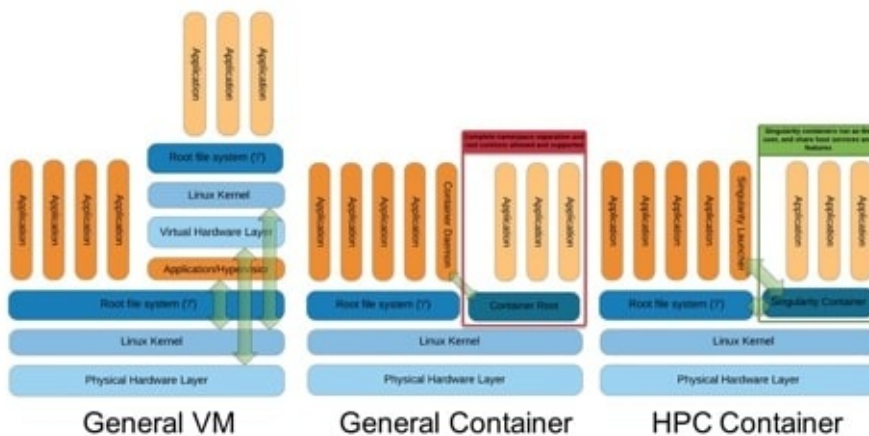
Figure 77: Singularity Container Architecture [69]

8.5.1.4 Hands-On Tutorials

The following tutorial includes links to asciinema video tutorials created by SDSC HPC Systems Manager, Trevor Cooper which allow you to see the console interactivity and output in detail. Look for the ▣ icon like the one shown to the right corresponding to the task you are currently working on.

8.5.1.5 Downloading & Installing Singularity

- Download & Unpack Singularity
- Configure & Build Singularity
- Install & Test Singularity

8.5.1.5.1 Download & Unpack Singularity

First we download and upack the source using the following commands (assuming your user name is `test_user` and you are working on your local computer with super user privileges):

```
[test_user@localhost ~]$ wget https://github.com/singularityware/singularity/
releases/download/2.5.1/singularity-2.5.1.tar.gz tar -zxf singularity-2.5.1.tar.gz
```

▣ [Singularity - download source and unpack in VirtualBox VM (CentOS 7)](#)

If the file is successfully extracted, you should be able to view the results:

```
[test_user@localhost ~]$ cd singularity-2.5.1/
[test_user@localhost singularity-2.5.1]$ ls
```

8.5.1.5.2 Configure & Build Singularity

[📹 Singularity - configure and build in VirtualBox VM (CentOS 7)](#)

Next we configure and build the package. To configure, enter the following command (we will leave out the command prompts):

```
./configure
```

To build, issue the following command:

```
make
```

This may take several seconds depending on your computer.

8.5.1.5.3 Install & Test Singularity

[📹 Singularity - install and test in VirtualBox VM (CentOS 7)](#)

To complete the installation enter:

```
sudo make install
```

You should be prompted to enter your admin password.

Once the installation is completed, you can check to see if it succeeded in a few different ways:

```
which singularity singularity -version
```

You can also run a selftest with the following command:

```
singularity selftest
```

The output should look something like:

```
+ sh -c test -f /usr/local/etc/singularity/singularity.conf (retval=0) OK  + test -u
/usr/local/libexec/singularity/bin/action-suid (retval=0) OK  + test -u /usr/local/libexec/singularity/bin/create-suid
(retval=0) OK  + test -u /usr/local/libexec/singularity/bin/expand-suid (retval=0) OK  + test -u
/usr/local/libexec/singularity/bin/export-suid (retval=0) OK  + test -u /usr/local/libexec/singularity/bin/import-suid
(retval=0) OK  + test -u /usr/local/libexec/singularity/bin/mount-suid (retval=0) OK
```

8.5.1.6 Building Singularity Containers

The process of building a Singularity container consists of a few distinct steps as follows.

- Upgrading Singularity (if needed)
- Create an Empty Container
- Import into Container
- Shell into Container
- Write into Container
- Bootstrap Container

We will go through each of these steps in detail.

8.5.1.6.1 Upgrading Singularity

We recommend building containers using the same version of Singularity, 2.5.1, as exists on Comet. This is a 2 step process.

Step 1: run the next script to remove your existing Singularity:

```bash
#!/bin/bash
#
# A cleanup script to remove Singularity

sudo rm -rf /usr/local/libexec/singularity
sudo rm -rf /usr/local/etc/singularity
sudo rm -rf /usr/local/include/singularity
sudo rm -rf /usr/local/lib/singularity
sudo rm -rf /usr/local/var/lib/singularity/
sudo rm /usr/local/bin/singularity
sudo rm /usr/local/bin/run-singularity
sudo rm /usr/local/etc/bash_completion.d/singularity
sudo rm /usr/local/man/man1/singularity.1
```

Step 2: run the following script to install Singularity 2.5.1:

```bash
#!/bin/bash
#
# A build script for Singularity (http://singularity.lbl.gov/)

declare -r SINGULARITY_NAME='singularity'
declare -r SINGULARITY_VERSION='2.5.1'
declare -r SINGULARITY_PREFIX='/usr/local'
declare -r SINGULARITY_CONFIG_DIR='/etc'

sudo apt update
sudo apt install python dh-autoreconf build-essential debootstrap

cd ../
tar -xzvf "${PWD}/tarballs/${SINGULARITY_NAME}-${SINGULARITY_VERSION}.tar.gz"
cd "${SINGULARITY_NAME}-${SINGULARITY_VERSION}"
./configure --prefix="${SINGULARITY_PREFIX}" --sysconfdir="${SINGULARITY_CONFIG_DIR}"
make
```

```
sudo make install
```

8.5.1.7 Create an Empty Container

[Singularity - create container](#)

To create an empty Singularity container, you simply issue the following command:

```
singularity create centos7.img
```

This will create a CentOS 7 container with a default size of ~805 Mb. Depending on what additional configurations you plan to make to the container, this size may or may not be big enough. To specify a particular size, such as ~4 Gb, include the -s parameter, as shown in the following command:

```
singularity create -s 4096 centos7.img
```

To view the resulting image in a directory listing, enter the following:

```
ls
```

8.5.1.8 Import Into a Singularity Container

[Singularity - import Docker image](#)

Next, we will import a Docker image into our empty Singularity container:

```
singularity import centos7.img docker://centos:7
```

8.5.1.9 Shell Into a Singularity Container

[Singularity - shell into container](#)

Once the container actually contains a CentOS 7 installation, you can 'shell' into it with the following:

```
singularity shell centos7.img
```

Once you enter the container you should see a different command prompt. At

this new prompt, try typing:

```
whoami
```

Your user id should be identical to your user id outside the container. However, the operating system will probably be different. Try issuing the following command from inside the container to see what the OS version is:

```
cat /etc/*-release
```

8.5.1.10 Write Into a Singularity Container

[image] [Singularity - write into container](#)

Next, let's trying writing into the container (as root):

```
sudo /usr/local/bin/singularity shell -w centos7.img
```

You should be prompted for your password, and then you should see something like the following:

```
Invoking an interactive shell within the container...
```

Next, let's create a script within the container so we can use it to test the ability of the container to execute shell scripts:

```
vi hello_world.sh
```

The previous command assumes you know the vi editor. Enter the following text into the script, save it, and quit the vi editor:

```
#!/bin/bash echo "Hello, World!"
```

You may need to change the permissions on the script so it can be executable:

```
chmod +x hello_world.sh
```

Try running the script manually:

```
./hello_world.sh
```

The output should be:

```
Hello, World!
```

8.5.1.11 Bootstrapping a Singularity Container

[📄] [Singularity - bootstrapping a container](#)

Bootstrapping a Singularity container allows you to use what is called a 'definitions file' so you can reproduce the resulting container configurations on demand.

Let us say you want to create a container with Ubuntu, but you may want to create variations on the configurations without having to repeat a long list of commands manually. First, we need our definitions file. Given next is the contents of a definitions file which should suffice for our purposes.

```
Bootstrap: docker
From: ubuntu:latest
%runscript
exec echo "The runscript is the containers default runtime command!"

%files
/home/testuser/ubuntu.def /data/ubuntu.def

%environment
VARIABLE=HELLOWORLD
Export VARIABLE
%labels
AUTHOR testuser@sdsc.edu

%post
apt-get update && apt-get -y install python3 git wget
mkdir /data
echo "The post section is where you can install and configure your container."
```

To bootstrap your container, first we need to create an empty container.

```
singularity create -s 4096 ubuntu.img
```

Now, we simply need to issue the following command to configure our container with Ubuntu:

```
sudo /usr/local/bin/singularity bootstrap ./ubuntu.img ./ubuntu.def
```

This may take a while to complete. In principle, you can accomplish the same result by manually issuing each of the commands contained in the script file, but why do that when you can use bootstrapping to save time and avoid errors.

If all goes according to plan, you should then be able to shell into your new Ubuntu container.

8.5.1.12 Running Singularity Containers on Comet

Of course, the purpose of this tutorial is to enable you to use the San Diego Supercomputer Center's Comet supercomputer to run your jobs. This assumes you have an account on Comet already. If you do not have an account on Comet and you feel you can justify the need for such an account (i.e. your research is limited by the limited compute power you have in your government-funded research lab), you can request a 'Startup Allocation' through the XSEDE User Portal:

https://portal.xsede.org/allocations-overview#types-trial

You may create a free account on the XUP if you do not already have one and then proceed to submit an allocation request at the previously given link.

NOTE: SDSC provides a Comet User Guide to help get you started with Comet. Learn more about The San Diego Supercomputer Center at http://www.sdsc.edu.

This tutorial walks you through the following four steps towards running your first Singularity container on Comet:

- Transfer the Container to Comet
- Run the Container on Comet
- Allocate Resources to Run the Container
- Integrate the Container with Slurm
- Use existing Comet Containers

8.5.1.12.1 Transfer the Container to Comet

Singularity - transfer container to Comet

Once you have created your container on your local system, you will need to transfer it to Comet. There are multiple ways to do this and it can take a varying amount of time depending on its size and your network connection speeds.

To do this, we will use scp (secure copy). If you have a Globus account and your containers are more than 4 Gb you will probably want to use that file transfer method instead of scp.

Browse to the directory containing the container. Copy the container to your scratch directory on Comet. By issuing the following command:

```
scp ./centos7.img comet.sdsc.edu:/oasis/scratch/comet/test_user/temp_project/
```

The container is ~805 Mb so it should not take too long, hopefully.

8.5.1.12.2 Run the Container on Comet

 Singularity - run container on Comet

Once the file is transferred, login to Comet (assuming your Comet user is named `test_user`):

```
ssh test_user@comet.sdsc.edu
```

Navigate to your scratch directory on Comet, which should be something like:

```
[test_user@comet-ln3 ~]$ cd /oasis/scratch/comet/test_user/temp_project/
```

Next, you should submit a request for an interactive session on one of Comet's compute, debug, or shared nodes.

```
[test_user@comet-ln3 ~]$ srun --pty --nodes=1 --ntasks-per-node=24 -p compute -t 01:00:00 --wait 0 /bin/bash
```

Once your request is approved your command prompt should reflect the new node id.

Before you can run your container you will need to load the Singularity module (if you are unfamiliar with modules on Comet, you may want to review the Comet User Guide). The command to load Singularity on Comet is:

```
[test_user@comet-ln3 ~]$ module load singularity
```

You may issue the previous command from any directory on Comet. Recall that we added a `hello_world.sh` script to our centos7.img container. Let us try executing that script with the following command:

```
[test_user@comet-ln3 ~]$ singularity exec /oasis/scratch/comet/test_user/temp_project/singularity/centos7.img
/hello_world.sh
```

If all goes well, you should see *Hello, World!* in the console output. You might also see some warnings pertaining to non-existent bind points. You can resolve

this by adding some additional lines to your definitions file before you build your container. We did not do that for this tutorial, but you would use a command like the following in your definitions file:

```
# create bind points for SDSC HPC environment mkdir /oasis /scratch/ /comet /temp_project
```

You will find additional examples located in the following locations on Comet:

```
/share/apps/examples/SI2017/Singularity
```

and

```
/share/apps/examples/SINGULARITY
```

8.5.1.12.3 Allocate Resources to Run the Container

[Singularity - allocate resources to run container](#)

It is best to avoid working on Comet's login nodes since they can become a performance bottleneck not only for you but for all other users. You should rather allocate resources specific for computationally-intensive jobs. To allocate a 'compute node' for your user on Comet, issue the following command:

```
[test_user@comet-ln3 ~]$ salloc -N 1 -t 00:10:00
```

This allocation requests a single node (-N 1) for a total time of 10 minutes (-t 00:10:00). Once your request has been approved, your computer node name should be displayed, e.g. comet-17-12.

Now you may login to this node:

```
[test_user@comet-ln3 ~]$ ssh comet-17-12
```

Notice that the command prompt has now changed to reflect the fact that you are on a compute node and not a login node.

```
[test_user@comet-06-04 ~]$
```

Next, load the Singularity module, shell into the container, and execute the `hello_world.sh` script:

```
[test_user@comet-06-04 ~]$ module load singularity [test_user@comet-06-04 ~]$ singularity shell centos7.img
[test_user@comet-06-04 ~]$ ./hello_world.sh
```

If all goes well, you should see *Hello, World!* in the console output.

8.5.1.12.4 Integrate the Container with Slurm

 [Singularity - run container on Comet via Slurm](#)

Of course, most users simply want to submit their jobs to the Comet queue and let it run to completion and go on to other things while waiting. Slurm is the job manager for Comet.

Given next is a job script (which we will name `singularity_mvapich2_hellow.run`) which will submit your Singularity container to the Comet queue and run a program, hellow.c (written in C using MPI and provided as part of the examples with the mvapich2 default installation).

```
#!/bin/bash `` #SBATCH --job-name="singularity_mvapich2_hellow"  #SBATCH --output="singularity_mvapich2_hellow.%j.out"
#SBATCH --error="singularity_mvapich2_hellow.%j.err"  #SBATCH --nodes=2  #SBATCH --ntasks-per-node=24  #SBATCH --

time=00:10:00  #SBATCH --export=all module load mvapich2_ib singularity

CONTAINER=/oasis/scratch/comet/$USER/temp_project/singularity/centos7-mvapich2.img

mpirun singularity exec ${CONTAINER} /usr/bin/hellow
```

The previous script requests 2 nodes and 24 tasks per node with a wall time of 10 minutes. Notice that two modules are loaded (see the line beginning with 'module'), one for Singularity and one for MPI. An environment variable 'CONTAINER' is also defined to make it a little easier to manage long reusable text strings such as file paths.

You may need to add a line specifying with allocation to be used for this job. When you are ready to submit the job to the Comet queue, issue the following command:

```
[test_user@comet-06-04 ~]$ sbatch -p debug ./singularity_mvapich2_hellow.run
```

To view the status of your job in the Comet queue, issue the following:

```
[test_user@comet-06-04 ~]$ squeue -u test_user
```

When the job is complete, view the output which should be written to the output file `singularity_mvapich2_hellow.%j.out` where `%j` is the job ID (let's say the job ID is 1000001):

```
[test_user@comet-06-04 ~]$ more singularity_mvapich2_hellow.1000001.out
```

The output should look something like the following:

```
Hello world from process 28 of 48
Hello world from process 29 of 48
Hello world from process 30 of 48
Hello world from process 31 of 48
Hello world from process 32 of 48
Hello world from process 33 of 48
Hello world from process 34 of 48
Hello world from process 35 of 48
Hello world from process 36 of 48
Hello world from process 37 of 48
Hello world from process 38 of 48
```

8.5.1.12.5 Use Existing Comet Containers

SDSC User Support staff, Marty Kandes, has built several custom Singularity containers designed specifically for the Comet environment.

[Learn more about these containers for Comet](#).

An easy way to pull images from the singularity hub on comment is provided in the next video:

📹 [Singularity - pull from singularity-hub on Comet](#)

Comet supports the capability to pull a container directly from any properly configured remote singularity hub. For example, the following command can pull a container from the hpcdevops singularity hub straight to an empty container located on Comet:

```
comet$ singularity pull shub://hpcdevops/singularity-hello-world:master
```

The resulting container should be named something like singularity-hello-world.img.

Learn more about Singularity Hubs and container collections at:

https://singularity-hub.org/collections

That's it! Congratulations! You should now be able to run Singularity containers on Comet either interactively or through the job queue. We hope you found this tutorial useful. Please contact support@xsede.org with any questions you might have. Your Comet-related questions will be routed to the amazing SDSC

Support Team.

8.5.1.13 Using Tensorflow With Singularity

One of the more common advantages of using Singularity is the ability to use pre-built containers for specific applications which may be difficult to install and maintain by yourself, such as Tensorflow. The most common example of a Tensorflow application is character recognition using the MNIST dataset. You can learn more about this dataset at http://yann.lecun.com/exdb/mnist/.

XSEDE's Comet supercomputer supports Singularity and provides several pre-built container which run Tensorflow. Given next is an example batch script which runs a Tensorflow job within a Singularity container on Comet. Copy this script and paste it into a shell script named mnist_tensorflow_example.sb.

```
#!/bin/bash
#SBATCH --job-name="TensorFlow"
#SBATCH --output="TensorFlow.%j.%N.out"
#SBATCH --partition=gpu-shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=6
#SBATCH --gres=gpu:k80:1
#SBATCH -t 01:00:00
```

8.5.1.14 Run the job

```
module load singularity
singularity exec
/share/apps/gpu/singularity/sdsc_ubuntu_gpu_tflow.img lsb_release
-a
singularity exec
/share/apps/gpu/singularity/sdsc_ubuntu_gpu_tflow.img python -m
tensorflow.models.image.mnist.convolutional
```

To submit the script to Comet, first you'll need to request a compute node with the following command (replace account with your XSEDE account number):

```
[test_user@comet-ln3 ~]$ srun --account=your_account_code --partition=gpu-shared --gres=gpu:1 --pty --nodes=1 --ntasks-pe
```

To submit a job to the Comet queue, issue the following command:

```
[test_user@comet-06-04 ~]$ sbatch mnist_tensorflow_example.sb
```

When the job is done you should see an output file in your output directory containing something resembling the following:

```
Distributor ID: Ubuntu
Description: Ubuntu 16.04 LTS
```

```
Release: 16.04
Codename: xenial
^[[33mWARNING: Non existent bind point (directory) in container: '/scratch'
^[[0mI tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcurand.so locally
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0 with properties:
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID 0000:85:00.0
Total memory: 11.17GiB
Free memory: 11.11GiB
I tensorflow/core/common_runtime/gpu/gpu_init.cc:126] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:136] 0: Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:838] Creating TensorFlow device (/gpu:0) -> (device: 0, name: Tesla K80, pci bus id: 000
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
Initialized!
Step 0 (epoch 0.00), 40.0 ms
Minibatch loss: 12.054, learning rate: 0.010000
Minibatch error: 90.6%
Validation error: 84.6%
Step 100 (epoch 0.12), 12.6 ms
Minibatch loss: 3.293, learning rate: 0.010000
Minibatch error: 6.2%
Validation error: 7.0%

Step 8400 (epoch 9.77), 11.5 ms
Minibatch loss: 1.596, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Step 8500 (epoch 9.89), 11.5 ms
Minibatch loss: 1.593, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.8%
Test error: 0.9%
```

Congratulations! You have successfully trained a neural network to recognize ascii numeric characters.

8.6 EXERCISES ☁

E.Docker.1: MongoDB Container

> *Develop a docker file that uses the mongo distribution from Dockerhub and starts a MongoDB database on the regular port while communicating to your container.*
>
> *What are the parameters on the command line that you need to define?*

E.Docker.2: MongoDB Container with authentication

> *Develop a MongoDB container that includes an outhenticated user.*

You must use the cloudmesh.yaml file for specifying the information for the admin user and password.

1. *How do you add the user?*
2. *How do you start the container?*
3. *Showcase the use of the authentication with a simple script or pytest.*

You are allowed tou sue docker compose, but make sure you read the password ond username from the yaml file. YoU must not configure it by hand in the compose yaml file. You can use cloudmesh commands to read the username and password.

```
cms config value cloudmesh.data.mongo.MONGO_USERNAME
cms config value cloudmesh.data.mongo.MONGO_PASSWORD
```

E.Docker.3: Cloudmesh Container

In this assignment we will explore the use of two containers. We will be leveraging the asisgnment E.Docker.2.

First, you wil lstart the authenticated docker MongoDB container

You will be writing an additional dockerfile, that creates cloudmesh in a docker container. Upon start the parameter passed to the container will be executed in the container. You will use the .ssh and .cloudmesh directory from your native file system.

For hints, please look at

- *https://github.com/cloudmesh/cloudmesh-cloud/blob/master/docker/ubuntu-19.04/Dockerfile*
- *https://github.com/cloudmesh/cloudmesh-cloud/blob/master/docker/ubuntu-19.04/Makefile*

To jump start you try

```
make image
make shell
```

Explore! Understand what is done in the Makefile

Questions:

1. *How would you need to modify the Dockerfile to complete it?*
2. *Whay did we outcomment the MongoDB related tasks in the Dockerfile?*
3. *How do we need to establish communication to the MongoDB container*
4. *Could docker compose help, or would it be too complicated, e.g. what if the mongo container already runs?*
5. *Why would it be dangerous to store the cloudmesh.yaml file inside the container? Hint: DockerHub.*
6. *Why should you at this time not upload images to DockerHub?*

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:

1. *Create a Dockerfile that deploys hadoop in a container*
2. *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/current/hadoop-project-dist/hadoop-common/SingleCluster.html)

E.MultiNodeHadoop:

Setup a hadoop cluster in a distributed environment.

This includes:

1. *Create docker compose and Dockerfiles that deploys hadoop in kubernetes*
2. *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/hadoop-project-dist/hadoop-common/ClusterSetup.html*](https://hadoop.apache.org/docs/r3.0.0/hadoop-project-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.

9 NIST

9.1 NIST BIG DATA REFERENECE ARCHITECTURE ☁

🎓 Learning Objectives

- Obtain an overview of the NIST Big Data Refernce 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 work 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 chose 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.

9.1.1 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" [70]. 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" [70]. 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," [70] 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" [70];
- Stage 3: "Validate the NBD-RA by building Big Data general applications through the general interfaces.[70]"

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

9.1.2 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 to be 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 challange exists for data engineers to develop architectures and their deployment implications. The volume of data and the processing power needed to analysis 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 tha 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 posible. 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 todays 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 source.

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 adress 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.

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.

9.1.3 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 Ligo [71] and Atlas)

In addition to the horizontal scaling issues we also have to worry about the *vertical scaling* issues, this is how the overall sytem architecture fits together to adress 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 weel 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 wile 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.

Naturally such resources can be used not only for numerical intensif computations but also for big data applications as the Physics community has demonstrated.

9.1.4 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.

9.1.5 Reference Architecture

Next we present the Big data reference architecture. It is Depicted in Figure 78. 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
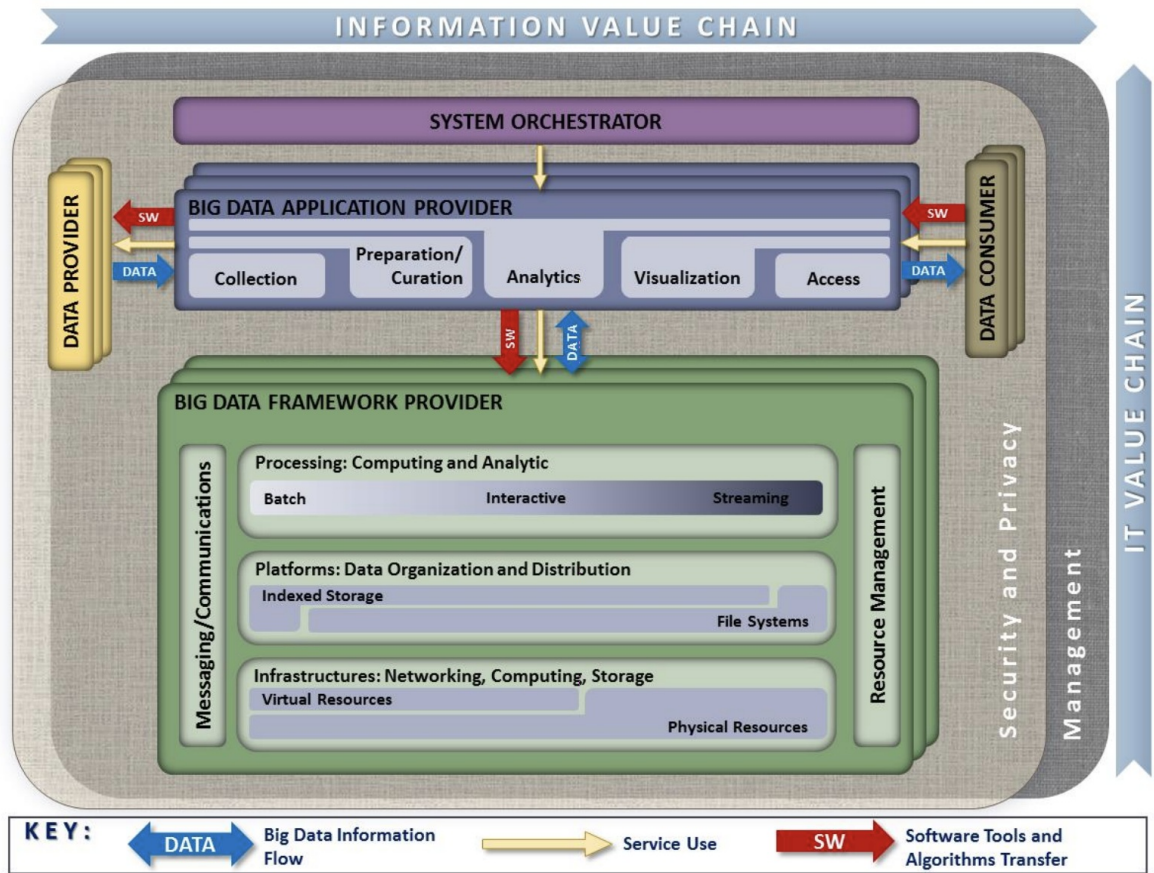
Figure 78: NIST-BDRA (see Volume 2)

While looking at the actors depicted in Figure 79 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.
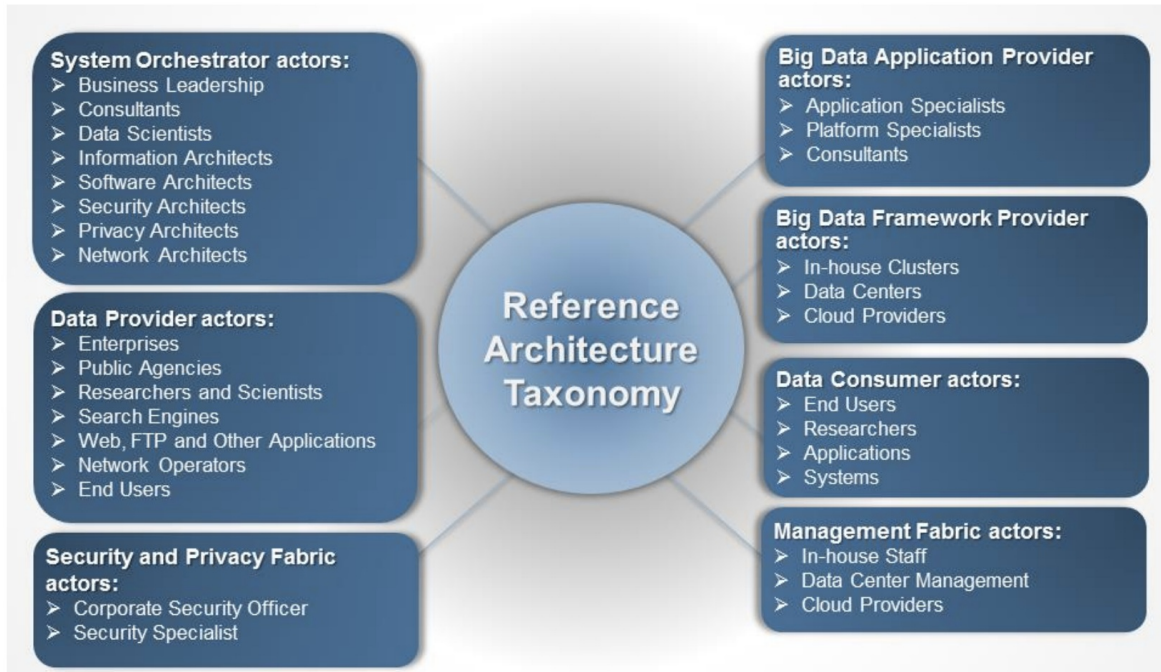
Figure 79: NIST Roles (see Volume 2)

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

9.1.6 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 layer while resource management is used to address efficiency.

Frameworks such as Spark and Hadoop include components form 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.

9.1.7 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.

9.1.8 Fabric

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

9.1.9 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.

10 AI

10.1 ARTIFICIAL INTELLIGENCE SERVICE WITH REST ⏺ ☁

⏺ {#sec:ai}

10.1.1 Unsupervised Learning

Keywords: clustering, kNN, Markov Model

Unsupervised learning is a learning method when the training data is not labeled. This problem could be more challenging because we are not supposed to have pre-knowledge and find patterns from the data.

Unsupervised learning could be very computing intensive and it has very complicated math principles, but very useful. In this chapter, we will illustrate some most popular unsupervised learning algorithms and raise examples how we apply them, which includes KMeans, k-NN, Markov Model and others.

It is important to know that unsupervised learning is just a way how we look at the problem. Each algorithm is just an example on how we solve a particular problem. Before you apply an algorithm, attention should be given to the reason why we apply a specific algorithm.

10.1.2 KMeans

KMeans is one of the most straight forward unsupervised learning algorithms.

slides[AI (40) Unsupervised Learning](AI (40) Unsupervised Learning)

10.1.3 Lab:Practice on AI

Keywords: Docker, REST Service, Spark

slides Practice on AI (40) REST services

10.1.4 k-NN

k-NN is a non-parametric statistical method meaning there is no assumption made about the distribution of the data. Additionally the distribution is not assumed to be fixed i.e. the distribution may change through time. These relaxed assumptions make non-parametric tests extremely valuable when applied to real-world data as a vast majority of real-world data have dynamic distributions though time, climate data is an obvious example. Non-Parametric data is often ordinal which means the variables have an inherent categorical order with unknown distances between the categories. A common example of a non-parametric statistical test is the sign test where values are assigned a positive or negative sign based on being above or below the median. In k-NN predictions are made about unknown values by matching the unknown values with similar known values. Naturally the determination of 'similar' is of fundamental importance. This is done through the application of the Euclidean distance calculation given by Equation 1.

$$d(\mathbf{i}, \mathbf{j}) = d(\mathbf{j}, \mathbf{i}) = \sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2 + \dots (i_n - j_n)^2} = \sqrt{\sum_{n=1}^{n}(i_n - j_n)^2}$$

To illustrate an example of calculating *similarity* using Equation 1 it can be determiend if a car is fast or not by using the data in Table 1. Lets pretend we know nothing about cars and are asked if we think a Chevy Corvette is fast or not.

Car make and model with associated horsepower, whether the vehicle has a racing stripe and if the author thinks the car is fast or not Table 1.

Table 1: Car Data

Car Name	Horsepower (HP)	Racing Stripe (Yes or No)	Fast (Yes or No)
Toyota Prius	120	0	0
Tesla Roadster	288	0	1
Bugatti Veyron	1200	1	1
Honda Civic	158	1	0

| Lamborghini Aventador | 695 | 1 | 1 |

Now lets say our friend wants to know if a Ford Mustang with a racing stripe is fast or not. This particular friend knows nothing about cars so decided to put analytics to work. Since a Mustang has roughly 300 horse power the closet car in our dataset to this is the Tesla Roadster and since the Tesla is fast we would predict the Mustang to be fast. Remember this is completely dependent on the authors initial classification of whether a car is fast or not. Clearly the Lamborghini and the Bugatti are fast but maybe the Tesla is not fast therefore giving an incorrect answer. An example using the Mustang and the Tesla is given in the next calculation:

$$d(\mathbf{i,j}) = \sqrt{(300-288)^2 + (1-0)^2} = 12.04$$

We were able to determine the closest, or first nearest neighbor by inspection of this data, however with a more robust dataset this may not be the case. In these situations to find the nearest neighbor the Euclidean distance is calculated for every unique row entry and then ordered from smallest to largest distances, naturally the smallest distances are the most similar. You may notice that the values of horsepower are significantly larger in magnitude than the values associated with racing stripes. This could be problematic in many real world scenarios where the columns associated with large values do not have as direct of an impact as horsepower does on the variable we are trying to predict–a car being fast. In the case where each column value has equal predictive power data normalization should be performed. This is the process of centering each column to a mean of zero (0) and a standard deviation of one (1). This is done by taking the column means and subtracting the column means from each column entry and dividing by the column standard deviation.

Determine for yourself if we use 2 nearest neighbors what the prediction about the Mustang would be given the data provided what about 3, 4 nearest neighbors? What is the maximum number of k-nearest neighbors we could have given the dataset in [Table 1](#) ?

Calculate the Euclidean Distances for all five row entries with respect to the Mustang.

Normalize the data and recalculate the first and second nearest neighbors with respect to the Mustang. Does anything change?

In order to see k-NN in action we will look at an in depth example using a dataset from the National Basketball Associated (NBA) from 2013, naturally there are more up to date datasets but as sports analytics becomes a significant market more and more data is becoming proprietary. This example will pick an NBA player and determine the most similar NBA player in the dataset to the selected player using k nearest neighbors. The following is set up for you to execute in a python command prompt line by line for instructional purposes.

```python
# This code was adopted from Dataquest - K nearest neighbors in Python:
# Written by: Vik Paruchuri
import pandas
import math
with open("/path/to/the/nba_2013.csv", 'r') as csvfile:
    nba = pandas.read_csv(csvfile)
```

The previous portion of code uses pandas to open the downloaded csv file and name it nba, naturally you could name the file anything. If you want to view the columns in the csv file the following command can be used.

```python
print(nba.columns.values)
```

Now we need to select a player from the dataset, we will then determine the most similar player to our selected player. Analysis like this is becoming more and more prevalent in professional sports due to the large amounts of money invested in players. Scouts may use this type of analysis to determine who a given prospect is most similar too. This following bit of code selects a player from the dataset. Notice that the column player is first selected followed by the player name.

```python
selected_player = nba[nba["player"] == "LastName FirstName"].iloc[0]
```

The next step is to remove any non-numeric columns from our analysis since we are using the Euclidean distance to calculate proximity and strings can not be evaluated in such a way. One thing you can do if you have columns that have values like yes and no is assign zeros and ones accordingly. In our case we will only select the columns with numeric values.

```python
numeric_columns = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p',
'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.',
'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf', 'pts']
```

We now have everything we need to calculate the Euclidean distance, there are built in functions available in python to calculate this however we will define our own as it is a straight forward computation. It is also good practice to define your own functions whenever possible.

```python
def euclidean_distance(row):
    """
    Define our own Euclidean distance function
    """
    euc_distance = 0
    for k in numeric_columns:
        euc_distance += (row[k] - selected_player[k]) ** 2
    return math.sqrt(euc_distance)
```

Applying our function using the following command will determine the Euclidean distance between the selected player and all other players in the dataset.

```python
selected_player_distance = nba.apply(euclidean_distance, axis=1)
```

For sake of argument we will assume that all the data columns have equal predictive capabilities so we wish to normalize. This will often be the case with sports statistics as total points and field goal percentage vary in magnitude significantly but total points does not necessarily hold more predictive power than field goal percentage. In order to normalize we again most only select the the numeric columns and text columns can not be normalized in the way described previously.

```python
nba_numeric = nba[numeric_columns]
#apply normalization formula using built in python math
#functions for the mean and standard deviation
nba_normalized = (nba_numeric - nba_numeric.mean()) / nba_numeric.std()
```

We can now use built in functions to calculate the nearest neighbors in order to compare to our results attained from the previous exercise. In case you did not notice the `selected_player_distance` array is an array that lists all the Euclidean distances. We will use this later to see if the same result is obtained by using the built in functions. First we will import the necessary libraries shown in the next code.

```python
from scipy.spatial import distance
```

If you inspected the the `selected_player_distance` array you would have noticed that there were several NaN's present this was due to having an incomplete dataset and must be avoided. The following bit of code will replace all NA entries with zeros.

```python
nba_normalized.fillna(0, inplace=True)
```

Using the built in Euclidean distance to determine the Euclidean distances of all players in the data set to our selected player.

```
player_normalized = nba_normalized[nba["player"] == "LastName FirstName"]
euclidean_distances = nba_normalized.apply(lambda row:
    distance.euclidean(row, player_normalized), axis=1)
```

Here we create a data frame to hold the distances and then sort the values from lowest to highest. Since our player will naturally be in the dataset the selected player will be the lowest value, therefore the second lowest value is associated with the player most closely related to our selected player.

```
distance_frame = pandas.DataFrame(data={"dist": euclidean_distances,
  "idx": euclidean_distances.index})
distance_frame.sort_values("dist", inplace=True)
second_smallest = distance_frame.iloc[1]["idx"]
most_similar_to_player = nba.loc[int(second_smallest)]["player"]
```

In the python prompt type:

```
most_similar_to_player
```

The most similar player to your selected player should appear.

10.1.5 Machine Learning and Cloud Services

10.1.5.1 Introduction and Regression

This video lecture covers logistic and linear regression models in additon to clustering models. The algorithims for the three methods are formalized and solutions are presented. Additionally visulization techinqies are introduced including WebPlotviz and matplotlib.

[Introduction to Machine Learning for Cloud Services and Regression 10:55](#)

10.1.5.2 K-means Clustering

Video lecture and slides cover an introduction to K-means clusters.

[K-means Clusters 17:15](#)

10.1.5.3 Visulization

Video lecture and slides cover data visulaitzation techinques using state of the science tools like WebPlotViz.

📹 [Data Visualization 30:10](#)

10.1.5.4 Clustering Examples

Video lecture and slides cover clustering examples.

📹 [Examples of Clustering 5:48](#)

10.1.5.5 General Clustering with Examples

Video lecture and slides take a generalized apprach to clustering with examples from Dr. Geoffery Fox's research.

📹 [General Clustering and Research Examples 22:28](#)

10.1.5.6 In Depth Example with four centers

Video lecture and slides use 1000 data points and four artifical centers to provide and in depth example of clustering. Code provided.

📹 [Example with four centers 20:53](#)

10.1.5.7 Parallel Computing and K-means

Video lecture and slides discuss parallel computing using K-means as an example of how to accelerate time to completion by exploiting modern computing hardware.

📹 [Parallel Computing and K-means](#)

10.1.6 Example Project with SVM

The following code is set up as an example project and will show how to use a RESTful service to download data. Additionally the differences between a dynamic and static API will be showcased. First we begin by importing the appropriate libraries.

```
import requests
from flask import Flask
import numpy as np
from sklearn.externals.joblib import Memory
from sklearn.datasets import load_svmlight_file
from sklearn.svm import SVC
from os import listdir
from flask import Flask, request
```

Next we define three functions required to run this example: a function to download the data; a function to partition the data; and a function to get the data into the appropriate format once downloaded.

```
app = Flask(__name__)

def download_data(url, filename):
    r = requests.get(url, allow_redirects=True)
    open(filename, 'wb').write(r.content)

def data_partition(filename, ratio):
    file = open(filename,'r')
    training_file=filename+'_train'
    test_file=filename+'_test'
    data = file.readlines()
    count = 0
    size = len(data)
    ftrain =open(training_file,'w')
    ftest =open(test_file,'w')
    for line in data:
        if(count< int(size*ratio)):
            ftrain.write(line)
        else:
            ftest.write(line)
        count = count + 1

def get_data(filename):
    data = load_svmlight_file(filename)
    return data[0], data[1]
```

Defining the first API endpoint with the following lines of code will allow the user to expose the API. Prove this to yourself by opening a browser, preferably google, and following the url next to the code.

```
@app.route('/')
def index():
    return "Demo Project!"

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

Now Open the application in your browser with

```
http://127.0.0.1:5000/
```

The first API endpoint we will define is the endpoint to download the data,

which is done by the following lines of code. Note the url is hardcoded into this portion of the code as passing urls to an API is not good practice.

```python
@app.route('/api/download/data')
def download():
    url =
    'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/glass.scale'
    download_data(url=url, filename='iris.scale')
    return "Data Downloaded"
```

The following three api endpoints use the data partition and get data functions defined previously. The partition function splits the datasets into two sections–testing and training. In this example the testing portion of the dataset is 20 % and the training is 80 % of the dataset. Later we will explore how to make this part dynamic, allowing the user to choose the partitioning percentage.

```python
@app.route('/api/data/partition')
def partition():
    data_partition('iris.scale',0.8)
    return "Successfully Partitioned"

@app.route('/api/get/data/test')
def gettestdata():
    Xtest, ytest = get_data("iris.scale_test")

    return "Return Xtest and Ytest arrays"

@app.route('/api/get/data/train')
def gettraindata():
    Xtrain, ytrain = get_data("iris.scale_train")

    return "Return Xtrain and Ytrain arrays"
```

The last bit of code is the implementation of SVM into a RESTful API endpoint. Again this is static and all parameters needed to tune the algorithm are hardcoded. It will be worth your time to extrapolate the discussion about dynamic APIs in order to make these parameters tunable by the user through the url.

```python
@app.route('/api/experiment/svm')
def svm():
    Xtrain, ytrain = get_data("iris.scale_train")
    Xtest, ytest = get_data("iris.scale_test")

    clf = SVC(gamma=0.001, C=100, kernel='linear')
    clf.fit(Xtrain, ytrain)

    test_size = Xtest.shape[0]
    accuarcy_holder = []
    for i in range(0, test_size):
        prediction = clf.predict(Xtest[i])
        print("Prediction from SVM: "+str(prediction)+", Expected
        Label : "+str(ytest[i]))
        accuarcy_holder.append(prediction==ytest[i])

    correct_predictions = sum(accuarcy_holder)
    print(correct_predictions)
    total_samples = test_size
    accuracy =
    float(float(correct_predictions)/float(total_samples))*100
    print("Prediction Accuracy: "+str(accuracy))
```

```
        return "Prediction Accuracy: "+str(accuracy)
```

In order to run this you need to make a directory in a location of your choice and create a file called main.py that has the code included in it. Then simply type the following command in a terminal where you have navigated to the location of the directory that your created.

```
python main.py
```

A continuous version of main.py is provided next for ease of use. Please be careful when copying and pasting as additional characters may show up, this was noticed in the url sections.

```python
import requests
from flask import Flask
import numpy as np
from sklearn.externals.joblib import Memory
from sklearn.datasets import load_svmlight_file
from sklearn.svm import SVC
from os import listdir
from flask import Flask, request


app = Flask(__name__)


def download_data(url, filename):
    r = requests.get(url, allow_redirects=True)
    open(filename, 'wb').write(r.content)


def data_partition(filename, ratio):
    file = open(filename,'r')
    training_file=filename+'_train'
    test_file=filename+'_test'
    data = file.readlines()
    count = 0
    size = len(data)
    ftrain =open(training_file,'w')
    ftest =open(test_file,'w')
    for line in data:
        if(count< int(size*ratio)):
            ftrain.write(line)
        else:
            ftest.write(line)
        count = count + 1


def get_data(filename):
    data = load_svmlight_file(filename)
    return data[0], data[1]


@app.route('/')
def index():
    return "Demo Project!"


@app.route('/api/download/data')
def download():
    url =
    'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/glass.scale'
    download_data(url=url, filename='iris.scale')
    return "Data Downloaded"


@app.route('/api/data/partition')
def partition():
    data_partition('iris.scale',0.8)
    return "Successfully Partitioned"


@app.route('/api/get/data/test')
def gettestdata():
    Xtest, ytest = get_data("iris.scale_test")
```

```
    return "Return Xtest and Ytest arrays"

@app.route('/api/get/data/train')
def gettraindata():
    Xtrain, ytrain = get_data("iris.scale_train")

    return "Return Xtrain and Ytrain arrays"

@app.route('/api/experiment/svm')
def svm():
    Xtrain, ytrain = get_data("iris.scale_train")
    Xtest, ytest = get_data("iris.scale_test")

    clf = SVC(gamma=0.001, C=100, kernel='linear')
    clf.fit(Xtrain, ytrain)

    test_size = Xtest.shape[0]
    accuarcy_holder = []
    for i in range(0, test_size):
        prediction = clf.predict(Xtest[i])
        print("Prediction from SVM: "+str(prediction)+", Expected
        Label : "+str(ytest[i]))
        accuarcy_holder.append(prediction==ytest[i])

    correct_predictions = sum(accuarcy_holder)
    print(correct_predictions)
    total_samples = test_size
    accuracy =
    float(float(correct_predictions)/float(total_samples))*100
    print("Prediction Accuracy: "+str(accuracy))
    return "Prediction Accuracy: "+str(accuracy)

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

As mentioned previously these these are examples of static API endpoints. In many scenarios having a dynamic API would be preferred. Lets explore the data partition endpoint and modify the code for the static version to make a dynamic version. The next part is the function definition for the dynamic version of the data_partition function, and not much has changed. The only change made was that stings were appended to the testing and training file names for convenience. The ratio will match the user defined ratio entered through the url.

```
def data_partition(filename, ratio):
    file = open(filename,'r')
    training_file=filename+'_train_'+str(ratio)
    test_file=filename+'_test_'+ str(ratio)
    data = file.readlines()
    count = 0
    size = len(data)
    ftrain =open(training_file,'w')
    ftest =open(test_file,'w')
    for line in data:
        if(count< int(size*ratio)):
            ftrain.write(line)
        else:
            ftest.write(line)
        count = count + 1
```

Now for defining the endpoint, naturally it starts the same way as the static version however now we must add a part that allows for the user to enter values. This is done by use of brackets < text >.

```python
@app.route('/api/data/partition/<filename>/ratio/<ratio>')
def partition(filename,ratio):
    ratio = float(ratio)
    path='data/'+filename
    data_partition(path,ratio)
    return "Successfully Partitioned"
```

11 REFERENCES

[1] L. Richardson, "Beautiful soup python package overview." Web Page, 2019 [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[2] 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/

[3] Guru99, "Introduction to mongodb." Web Page, 2018 [Online]. Available: https://www.guru99.com/mongodb-tutorials.html#1

[4] MongoDB, "Https://www.mongodb.com/." Web Page, 2018 [Online]. Available: https://docs.mongodb.com/manual/introduction/

[5] 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

[6] 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

[7] 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/

[8] 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/

[9] 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/

[10] 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/

[11] MongoDB, *Security*, 4.0 ed. New York City, New York, United States: MongoDB Inc, 2008 [Online]. Available: https://docs.mongodb.com/manual/security/

[12] MongoDB, "MongoDB atlas." Web Page, 2018 [Online]. Available: https://www.mongodb.com/cloud/atlas

[13] I. MongoDB, "PyMongo 3.7.1 documentation." Web Page, 2008 [Online]. Available: https://api.mongodb.com/python/current/api

[14] A. J. J. Davis, "Announcing pymongo3." Web Page, Apr-2015 [Online]. Available: https://emptysqua.re/blog/announcing-pymongo-3/

[15] M. Dirolf, "PyMongo." Web Page, Jul-2018 [Online]. Available: https://github.com/mongodb/mongo-python-driver

[16] N. Leite, "MongoDB and python." Web Page, Mar-2015 [Online]. Available: https://www.slideshare.net/NorbertoLeite/mongodb-and-python

[17] 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/

[18] I. MongoDB, "Installing / upgrading." Web pages, 2008 [Online]. Available: http://api.mongodb.com/python/current/installation.html

[19] R. Python, "Introduction to mongodb and python." Web Page, 2016 [Online]. Available: https://realpython.com/introduction-to-mongodb-and-python/

[20] W3Schools, "Python mongodb create database." Web Page, 1999 [Online]. Available: https://www.w3schools.com/python/python_mongodb_create_db.asp

[21] I. MongoDB, "PyMongo 3.7.1 documentation." Web Page, 2008 [Online]. Available: https://api.mongodb.com/python/current/tutorial.html

[22] N. O'Higgins, *PyMongo & python.* O'Reilly, 2011 [Online]. Available: http://img105.job1001.com/upload/adminnew/2015-04-07/1428393873-MHKX3LN.pdf

[23] I. MongoDB, "PyMongo 3.7.1 documentation." Web Page, 2008 [Online]. Available: https://api.mongodb.com/python/current/examples/aggregation.html

[24] MongoDB, "PyMongo 3.7.2 documenation." Web Page, 2008 [Online]. Available: https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/

[25] MongoDB, "PyMongo 3.7.2 documenation." Web Page, 2008 [Online]. Available: https://docs.mongodb.com/manual/core/map-reduce/

[26] MongoDB, "PyMongo v2.0 documentation." Web Page, 2008 [Online]. Available: https://api.mongodb.com/python/2.0/examples/map_reduce.html

[27] MongoDB, "PyMongo 3.7.2 documenation." Web Page, 2008 [Online]. Available: https://api.mongodb.com/python/current/examples/copydb.html

[28] MongoEngine, "MongoEngine user documentation." Web Page, 2009 [Online]. Available: http://docs.mongoengine.org/

[29] Wikipedia, "Object-relational mapping." Web Page, May-2009 [Online]. Available: https://en.wikipedia.org/wiki/Object-relational_mapping

[30] MongoDB, "Flask-mongoengine." Web Page, 2008 [Online]. Available: http://docs.mongoengine.org/guide/defining-documents.html

[31] MongoEngine, "User guide: Document instances." Web Page, 2009 [Online]. Available: http://docs.mongoengine.org/guide/document-instances.html

[32] MongoEngine, "2.1 installing mongoengine." Web Page, 2009 [Online]. Available: http://docs.mongoengine.org/guide/installing.html

[33] MongoEngine, "2.2 connection to mongodb." Web Page, 2009 [Online]. Available: http://docs.mongoengine.org/guide/connecting.html

[34] MongoEngine, "User guide 2.5. Querying the database." Web Page, 2009 [Online]. Available: http://docs.mongoengine.org/guide/querying.html

[35] Wikipedia, "Flask (web framework)." Web Page, 2010 [Online]. Available: https://en.wikipedia.org/wiki/Flask_(web_framework)

[36] MongoDB, "Flask-pymongo." Web Page, 2008 [Online]. Available: https://flask-pymongo.readthedocs.io/en/latest/

[37] MongoDB, "Flask mongoalchemy." Web Page, 2008 [Online]. Available: https://pythonhosted.org/Flask-MongoAlchemy/

[38] MongoDB, "Flask-mongoengine." Web Page, 2008 [Online]. Available: http://docs.mongoengine.org/projects/flask-mongoengine/en/latest/

[39] Wikipedia, "Flask (web framework)." Web Page, Oct-2018 [Online]. Available: https://en.wikipedia.org/wiki/Flask_(web_framework)

[40] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures*, vol. 7. University of California, Irvine Doctoral dissertation, 2000.

[41] Wikipedia, "Representational state transfer." Web Page, 2019 [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer

[42] OpenAPI Initiative, "The openapi specification." Web Page [Online]. Available: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

[43] OpenAPI Initiative, "The openapi specification." Web Page [Online]. Available: https://github.com/OAI/OpenAPI-Specification

[44] RAML, "RAML version 1.0: RESTful api modeling language." Web Page [Online]. Available: https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md

[45] R. H. Kevin Burke Kyle Conroy, "Flask-restful." Web Page [Online]. Available: https://flask-restful.readthedocs.io/en/latest/

[46] E. O. Ltd, "Django rest framework." Web Page [Online]. Available: https://www.django-rest-framework.org/

[47] S. Software, "API development for everyone." Web Page [Online]. Available: https://swagger.io

[48] S. Software, "Swagger codegen documentation." Web Page [Online]. Available: https://swagger.io/docs/open-source-tools/swagger-codegen/

[49] A. Y. W. Hate, "OpenAPI.Tools." Web Page [Online]. Available: https://openapi.tools/

[50] "Hadoop mapreduce." Aug-2019 [Online]. Available: https://www.edureka.co/blog/mapreduce-tutorial/?utm_source=youtube&utm_campaign=mapreduce-tutorial-161216-wr&utm_medium=description

[51] "Hadoop mapreduce." Aug-2019 [Online]. Available: https://www.youtube.com/watch?v=SqvAaB3vK8U&list=WL&index=25&t=2547s

[52] "Apache mapreduce." Aug-2019 [Online]. Available: https://www.ibm.com/analytics/hadoop/mapreduce

[53] Wikipedia, "MapReduce." Aug-2019 [Online]. Available: https://en.wikipedia.org/wiki/MapReduce

[54] "Hadoop mapreduce." Aug-2019 [Online]. Available: https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm

[55] A. Khan, "Hadoop and spark." Aug-2019 [Online]. Available: https://www.quora.com/What-is-the-difference-between-Hadoop-and-Spark. [Accessed: 03-Sep-2017]

[56] "Apache spark vs hadoop mapreduce." Aug-2019 [Online]. Available: https://data-flair.training/blogs/apache-spark-vs-hadoop-mapreduce/

[57] Twister, "Twister2: Twister2 Big Data Hosting Environment: A composable framework for high-performance data analytics." Web Page, Feb-2017 [Online]. Available: https://twister2.gitbook.io/twister2/

[58] Twister, "Twister2: Twister2 Big Data Hosting Environment: A composable framework for high-performance data analytics." Web Page, Feb-2017 [Online]. Available: https://github.com/DSC-SPIDAL/twister2/

[59] Twister, "Twister2 word count example." Aug-2019.

[60] Twister, "Task examples." Web Page, Feb-2017 [Online]. Available: https://twister2.gitbook.io/twister2/examples/task_examples

[61] Twister, "Communication Model." Web Page, Feb-2017 [Online]. Available: https://twister2.gitbook.io/twister2/concepts/communication/communication-model

[62] S. Kamburugamuve *et al.,* "Twister: Net-communication library for big data processing in hpc and cloud environments," in *2018 ieee 11th international conference on cloud computing (cloud),* 2018, pp. 383–391.

[63] Twister2, "Kmeans performance comparison." Web Page, Jan-2019 [Online]. Available: https://twister2.gitbook.io/twister2/

[64] Twister, "Twister Examples." Web Page, Feb-2017 [Online]. Available: https://twister2.gitbook.io/twister2/examples

[65] Docker, "Overview of docker hub." Web Page, Mar-2018 [Online]. Available: https://docs.docker.com/docker-hub/

[66] 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

[67] Docker, "Repositories on docker hub." Web Page, Mar-2018 [Online]. Available: https://docs.docker.com/docker-hub/repos/

[68] 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](hub-b51fb545dd8e)

[69] G. M. Kurtzer, "Singularity Containers for Science." Presentation, Jan-2019 [Online]. Available: http://www.hpcadvisorycouncil.com/events/2017/stanford-workshop/pdf/GMKurtzer_Singularity_Keynote_Tuesday_02072017.pdf#43

[70] National Institute of Standars, "NIST big data public working group." Aug-2019 [Online]. Available: https://bigdatawg.nist.gov/

[71] LIGO, "Ligo data grid." Sep-2019 [Online]. Available: https://www.lsc-group.phys.uwm.edu/lscdatagrid/overview.html