DALHOUSIE UNIVERSITY

Faculty of Computer Science

CSCI 5409

Advanced Topics in Cloud Computing:

Report - Assignment 2

Feb. 28th, 2019

Submitted by:
Satya Kumar Itekela
Dalhousie ID: B00839907
Brightspace: st798799

Section A

Installing and Configuring Mesos

To install the Mesos server, I have used the Ubuntu server running on the AWS Platform. I used Ubuntu server 16.04 on AWS to install and configure Mesos.

EC2 Instance

1. Logged in to the AWS account and navigated to EC2.
2. Lanched EC2 instance and selected Ubuntu Server 16.04 LTS AMI.
3. Added Storage, Security Groups and launched EC2 instance.

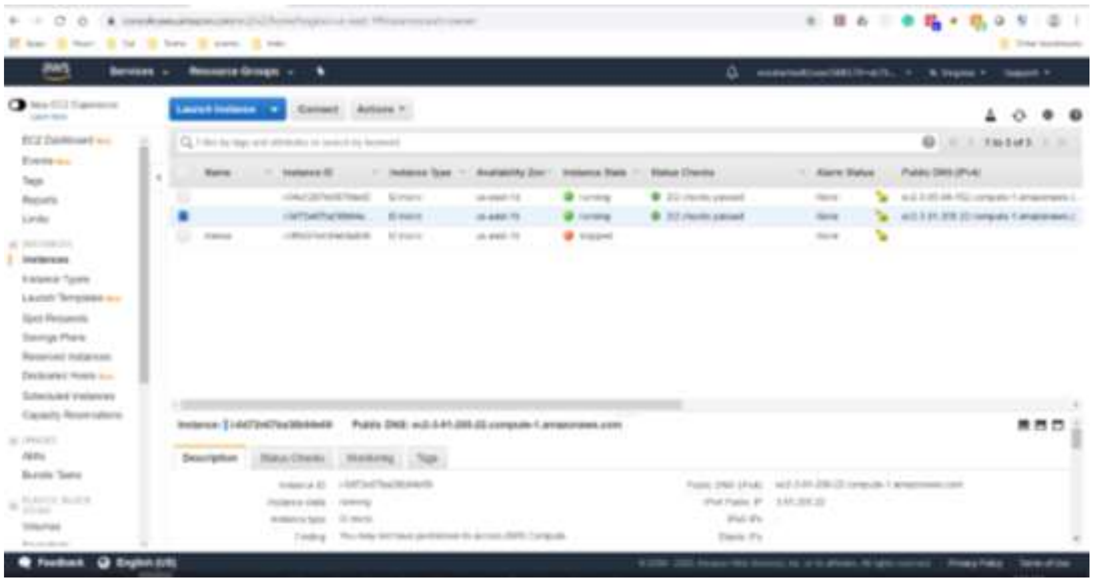


Figure 1EC2 Instance with ubuntu

The figure following is with the private DNS

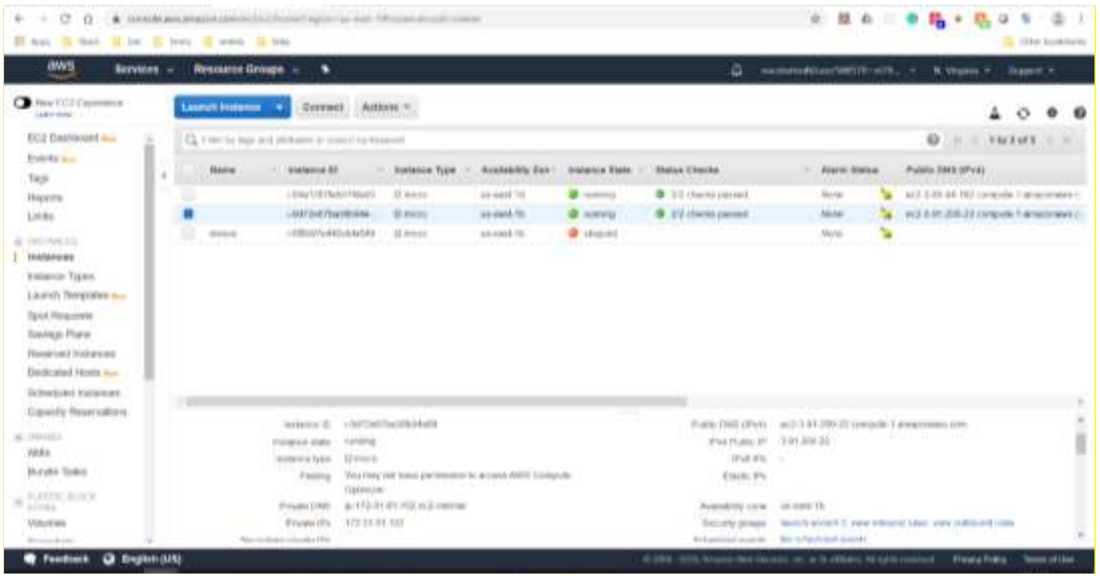


Figure 2 EC2 Instance with Private DNS

Installing Java8 and Mesos

1. Installed Java 8 using the tar file (latest JDK) downloaded from the Oracle official website.
2. Extracted the downloaded JDK using the following command [1]

   ```
   sudo tar -xzvf  ~/jdk-8u421-linux-x64.tar.gz
   ```

3. Edited the environment variables using the following commands [1]

   ```
   PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr
   /lib/jvm/jdk1.8.0_241/bin:/usr/lib/jvm/jdk1.8.0_241/db/bin:/usr/lib/jvm/jdk1.8.0_241/jre/bin"
   J2SDKDIR="/usr/lib/jvm/jdk1.8.0_241"
   J2REDIR="/usr/lib/jvm/jdk1.8.0_241/jre*
   JAVA_HOME="/usr/lib/jvm/jdk1.8.0_241"
   DERBY_HOME="/usr/lib/jvm/jdk1.8.0_241/db"
   ```

4. Updated the alternatives to inform Ubuntu using the following commands [1]

   ```
   sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.8.0_241/bin/java" 0
   sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/lib/jvm/jdk1.8.0_241/bin/javac" 0
   sudo update-alternatives --set java /usr/lib/jvm/jdk1.8.0_241/bin/java
   sudo update-alternatives --set javac /usr/lib/jvm/jdk1.8.0_241/bin/javac
   ```

5. Added OpenPGP key for Mesos packages from ubuntu server [2]

   ```
   sudo apt-key adv --keyserver keyserver.ubuntu.com --recv DF7D54CBE56151BF
   ```

6. Added Mesos repository to ubuntu [2]

   ```
   echo "deb http://repos.mesosphere.com/ubuntu xenial main" | sudo tee -a
   /etc/apt/sources.list.d/mesosphere.list
   ```

7. Updated the package indexes [2]

   ```
   sudo apt-get update
   ```

8. Installed Mesos. Zookeeper binaries [2]

   ```
   sudo apt-get -y install mesos
   ```

9. Navigated to the /etc/zookeeper/conf directory and edited the zoo.conf file and changed the server address to the Private DNS (ip-172-31-81-102.ec2.internal) of the launched EC2 instance.

10. Edited myid file to 1 that is the server id - using echo -n "1" > myid [2]

11. Navigated to the /etc/mesos directory and edited the zk file to point out the zookeeper instance

12. Navigated to the /etc/Mesos-master and edited the quorum file with an integer

13. Configured the Ip address and hostname of the Mesos master

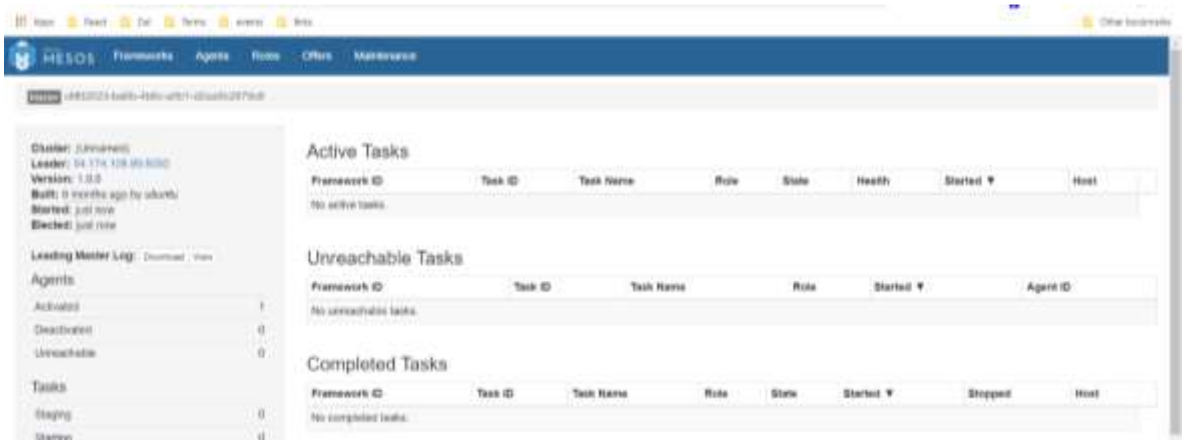
14. After configuring the necessary services, Started the services using the following commands [2]

```
service zookeeper start
service mesos-master start
service mesos-slave start
```

```
[illegible]
```

Figure 3 Services started on the EC2 instance

To access Mesos Ui, visited http://ec2-3-91-200-22.compute-1.amazonaws.com:5050/ URL



Cluster: (Unnamed)
Leader: [illegible]
Version: 1.0.0
Built: 8 months ago by ubuntu
Started: just now
Elected: just now

Leading Master Log: Download View

Agents

Activated	1
Deactivated	0
Unreachable	0

Tasks

Staging	0
Starting	0

Active Tasks

Framework ID	Task ID	Task Name	Role	State	Health	Started ▼	Host
No active tasks.							

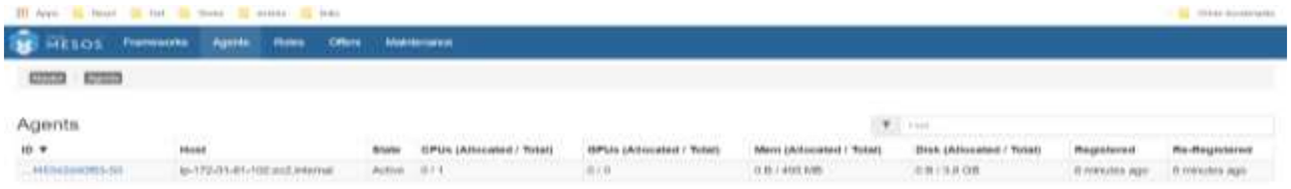
Unreachable Tasks

Framework ID	Task ID	Task Name	Role	Started ▼	Agent ID
No unreachable tasks.					

Completed Tasks

Framework ID	Task ID	Task Name	Role	State	Started ▼	Stopped	Host
No completed tasks.							

Figure 4 Mesos Master installed locally



Agents

ID ▼	Host	State	CPUs (Allocated / Total)	GPUs (Allocated / Total)	Mem (Allocated / Total)	Disk (Allocated / Total)	Registered	Re-Registered
[illegible]	ip-172-31-81-102.ec2.internal	Active	0 / 1	0 / 0	0 B / 493 MB	0 B / 3.8 GB	8 minutes ago	8 minutes ago

Figure 5 Mesos Agent running locally

Section B

Summary from “Exploring the Fairness and Resource Distribution in an Apache Mesos Environment" paper [3]

Apache Mesos is a cluster-wide resource manager that is commonly used in various clouds and data centres. It aims to provide high usage of clusters through resource co-scheduling and allocation based on Dominant Resource Fairness (DRF). For each application, DRF take the different types of resources needed into consideration and the amounts of each cluster resource that might be used. Mesos allocates resources and launch tasks on multiple frameworks like Apache Aurora, Mesosphere Marathon for container orchestration, and Chronos for cron jobs using its two-level scheduling mechanism. Weighted max-minute fairness is designed to provide fair share assurance policies such as max-min fairness and generalized variation. These will not work satisfactorily when the applications can co-scheduled to multi-resource types like memory, disc I / O, and network bandwidth with the same physical nodes.

Apache Mesos is comprised of three main components namely Mesos Master, Mesos Agent, and Mesos Framework. Initially, Mesos Agents frequently inform the Mesos Master about free resources that can be used to initiate tasks. After informing, Mesos master provides resources that are sorted out in the order of each share of resources. Finally, it launches the chosen agents and allocates the received resources to multiple tasks. Mesos Framework has two important modules namely Framework Scheduler, Executor. Scheduler is responsible for picking offers from the offers and creating a task map with offers, which uses Bin Packing and First Kit scheduling policies. These allocate resources to available frameworks based on the DRF fairness policy. An executor is notified when an agent node accepts frameworks tasks.

Mesos framework either accepts or declines the resources and allocates from a single node to competing users. The allocation module of master does not take into account of user requirements and the various attributes which have an impact on a fair number of resources include the interaction with Mesos resource offer cycles where a priority list is prepared for each offer cycle based on the dominant share, offer holding period where Mesos does not consume lengthy activities that is responsible for the other users, task arrival rate, and task duration.

On comparing the fairness with Marathon and Scylla frameworks, Marathon employs greedy resource consumption policies to obtain more resources, despite the fact that other frameworks waited for resources that reduced Scylla's fair resource allocation by 38 percent. Likewise, Aurora can not initiate more tasks because resources are less available compared to Aurora and Scylla, and the equal allocation of resources will be decreased by 89 percent. Scylla receives unfair resource allocation as it starves for a longer period of time. In conclusion, the DRF allocations for frameworks, like Aurora, Scylla and Marathon do not produce good fairness in Apache Mesos.

Insightful Comments

I have the following insights on the basis of my research on Apache Mesos, the biggest advantage of using Mesos is it provides efficient resources utilization and sharing across the multiple applications or frameworks. Based on the different requirements in different ways, Apache Mesos is able to manage various types of workloads. It is easy to deploy and manage various applications efficiently and can run more applications on a pool of nodes. The use of Apache Mesos can greatly benefit multi-node applications.

Section C

Installation of Marathon on Mesos cluster

1. Downloaded and Unpacked the latest Marathon release [4]
2. Environment variable MESOS_NATIVE_JAVA_LIBRARY has to be set that is available in the lib folder. [4]
3. Navigated to the marathon folder
4. Executed the following command to run the marathon

   nohup ./marathon --master zk://ip-172-31-81-102.ec2.internal:2181/mesos --zk zk://ip-172-31-81-102.ec2.internal:2181/marathon | sudo tee -a /var/log/marathon.log 2>&1 &

To access Marathon UI, visited the http://ec2-3-91-200-22.compute-1.amazonaws.com:8080/ url

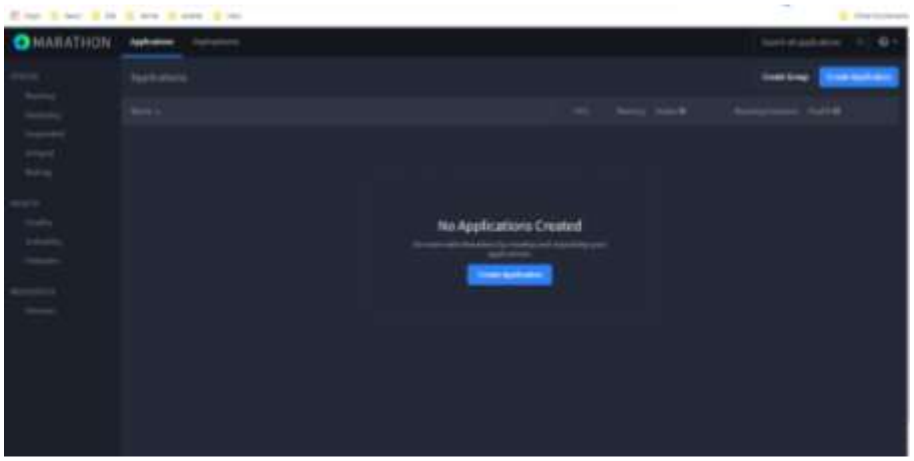
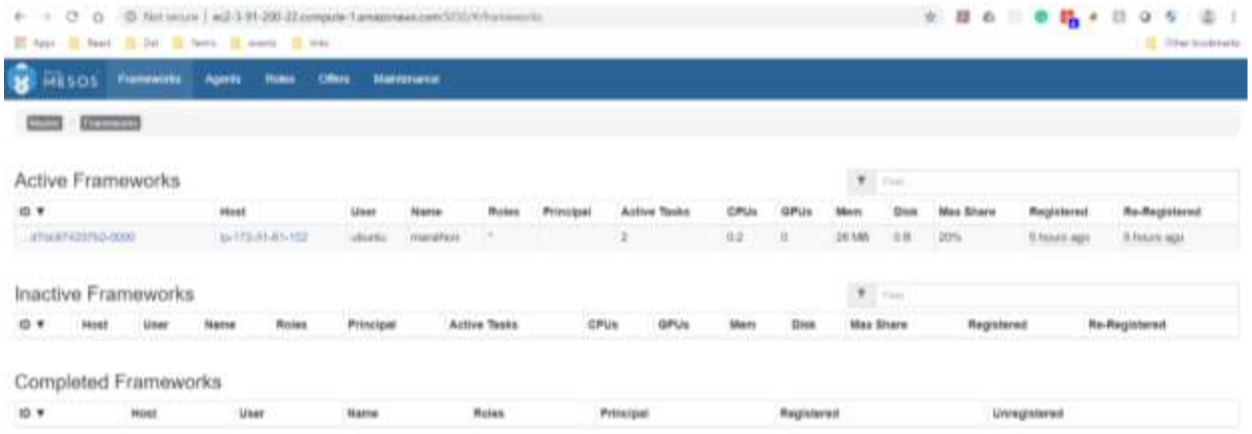


Figure 6 Marathon UI

Marathon framework registration with Mesos



Active Frameworks

ID ▼	Host	User	Name	Roles	Principal	Active Tasks	CPUs	GPUs	Mem	Disk	Max Share	Registered	Re-Registered
...d7bc87423750-0000	ip-172-31-81-102	ubuntu	marathon	*		2	0.2	0	26 MB	0 B	20%	9 hours ago	9 hours ago

Inactive Frameworks

ID ▼	Host	User	Name	Roles	Principal	Active Tasks	CPUs	GPUs	Mem	Disk	Max Share	Registered	Re-Registered

Completed Frameworks

ID ▼	Host	User	Name	Roles	Principal	Registered	Unregistered

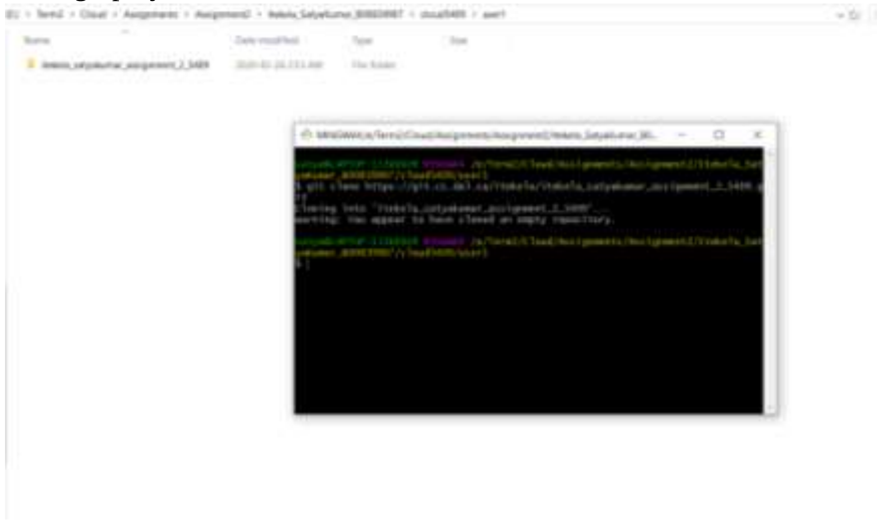
Figure 7 Marathon registered with Mesos

Section D

Repository Link

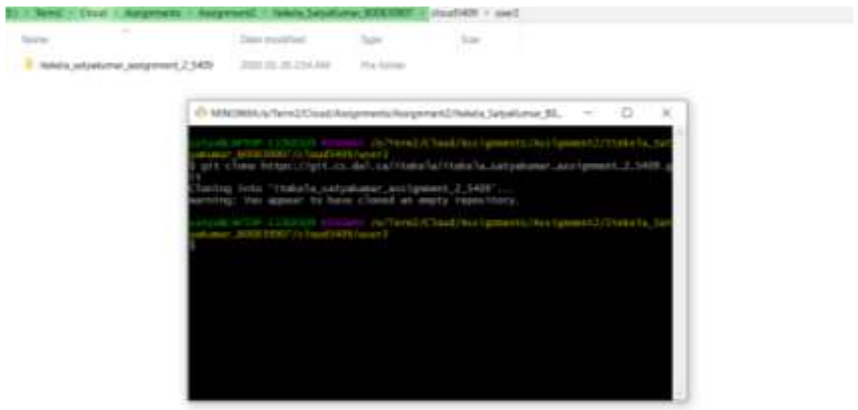
https://git.cs.dal.ca/itekela/itekela_satyakumar_assignment_2_5409

1. Created a git project itekela_satyakumar_assignment_2_5409
2. In the Local repository, I have created a folder cloud5409 and two folders user1 and user2 inside that folder.
3. I have cloned the git project in both users folders



```
[illegible] /e/Term2/Cloud/Assignments/Assignment2/Itekela_Sat
yakumar_[illegible]/cloud5409/user1
$ git clone https://git.cs.dal.ca/itekela/itekela_satyakumar_assignment_2_5409 [illegible]
Cloning into 'itekela_satyakumar_assignment_2_5409'...
warning: You appear to have cloned an empty repository.

[illegible] /e/Term2/Cloud/Assignments/Assignment2/Itekela_Sat
yakumar_[illegible]/cloud5409/user1
$
```

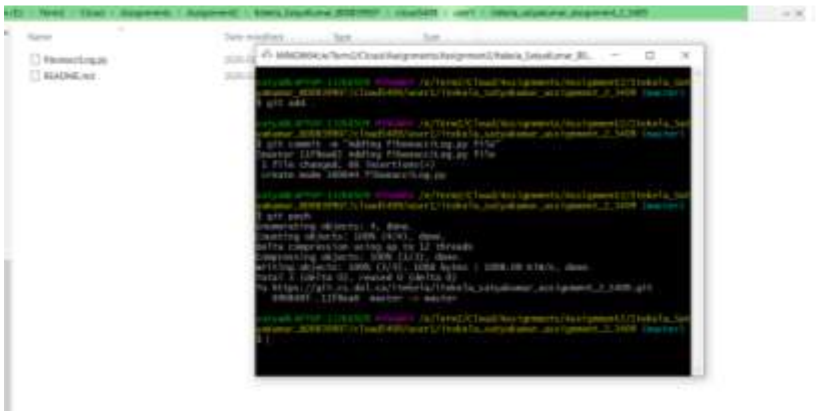
Figure 8 Cloning in User 1



```
[illegible] /e/Term2/Cloud/Assignments/Assignment2/Itekela_Sat
yakumar_[illegible]/cloud5409/user2
$ git clone https://git.cs.dal.ca/itekela/itekela_satyakumar_assignment_2_5409 [illegible]
Cloning into 'itekela_satyakumar_assignment_2_5409'...
warning: You appear to have cloned an empty repository.

[illegible] /e/Term2/Cloud/Assignments/Assignment2/Itekela_Sat
yakumar_[illegible]/cloud5409/user2
$
```

Figure 9 Cloning in User 2

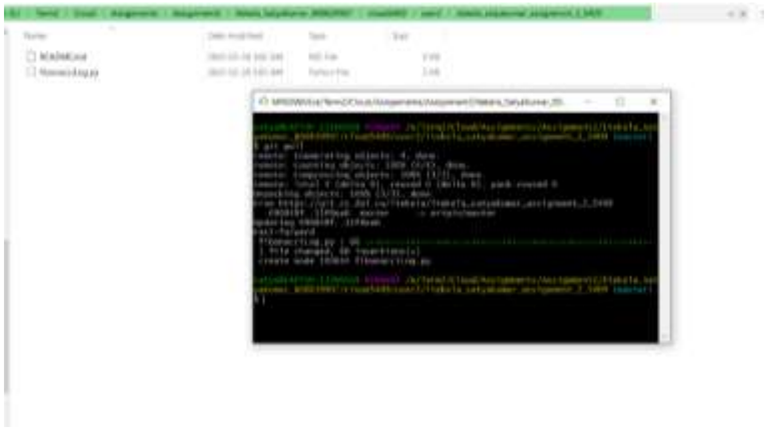
4. Added the Fibonacci file to the user 1 folder and add to the git repository



```
[illegible]
$ git add .

[illegible]
$ git commit -m [illegible]
[illegible]

[illegible]
$ git push
[illegible]

[illegible]
$ |
```

Figure 10 Adding the Fibonacci file to the repository

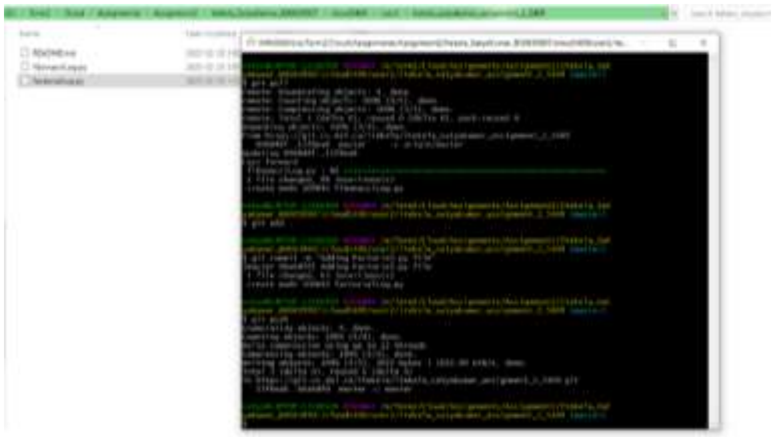
5. Received the file in the user 2 folder using pull command



```
[illegible]
$ git pull
[illegible]

[illegible]
$ |
```

Figure 11 Received the Fibonacci file to user2

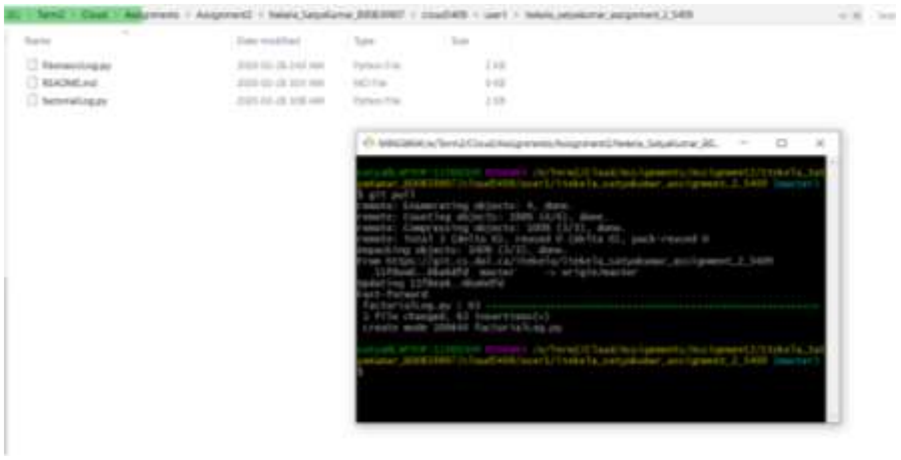
6. Adding new factorial file in the user 2 folder and adding the file to the repository



```
[illegible]
$ git pull
[illegible]

[illegible]
$ git add .

[illegible]
$ git commit -m [illegible]
[illegible]

[illegible]
$ git push
[illegible]

[illegible]
```

Figure 12 Adding Factorial file to the repository from user2

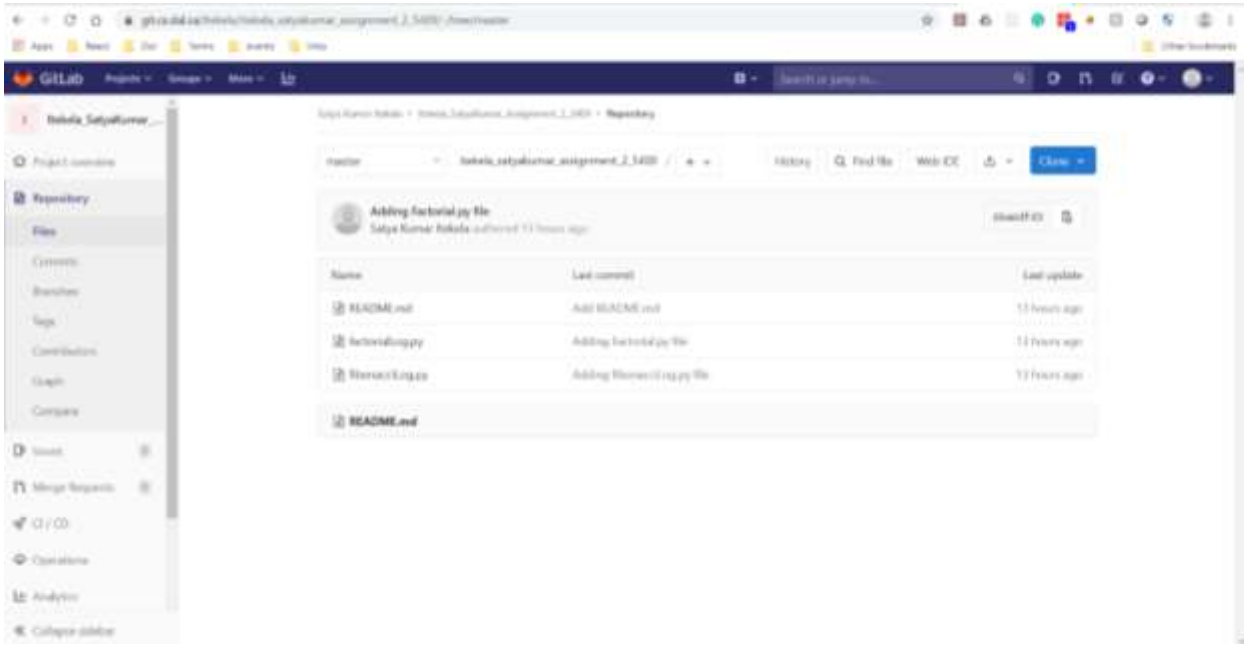
7. Fetching the factorial file in the user 1 folder



```
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://git.cs.dal.ca/[illegible]/[illegible]_satyakumar_assignment_2_5409
   [illegible]..[illegible]  master     -> origin/master
Updating [illegible]..[illegible]
Fast-forward
 factorialLog.py | 43 ++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 43 insertions(+)
 create mode 100644 factorialLog.py
```

Figure 13 Fetching the factorial file to the user 1 folder

Git commits in the repository



Adding factorial.py file
Satya Kumar Itekela authored 13 hours ago

Name	Last commit	Last update
README.md	Add README.md	13 hours ago
factorialLog.py	Adding factorial.py file	13 hours ago
fibonacciLog.py	Adding fibonacciLog.py file	13 hours ago

Figure 14 Git commits in the repository

Section E

Scenario

To perform two tasks S1 (factorial) and S2 (Fibonacci) parallelly which will read the input text file. The input file contains 100 random numbers. These two tasks perform the computations and generate the log that contains request ID, time, N, a result for each respective N value.

Source Code and Libraries

1. Folder cloud5409_SectionE contains factorialLog.py, fibonacciLog.py, factorialLog.csv, fibonacciLog.csv, input.txt, randomInt.py, readme.txt, factorial.png, Fibonacci.png
2. Libraries used – matplotlib, random, pandas and time

The whole scenario is divided into different files to solve the program

randomInt.py – contains a program that generates 100 random integers and stores in the text file

input.txt – contains 100 random numbers that are generated using random function.

factorialLog.py – contains the factorial function that takes the n value and returns it factorial, calculates the time taken for each request and returns the log into CSV that contains RequestId that are generated randomly using the random function, time taken to process for each request, N value and the result that is processed. It also generates a graph between each N value and the time taken to execute the request.

fibonacciLog.py – contains the factorial function that takes the n value and returns it Fibonacci series, calculates the time taken for each request and returns the log into CSV that contains RequestId that are generated randomly using the random function, time taken to process for each request, N value and the result that is processed. It also generates a graph between each N value and the time taken to execute the request.

Fibonacci.csv, factorial.csv – Stores requestId, time, N, the result for each request Id.

In Marathon two tasks are created factorial and Fibonacci as shown in the figure below.

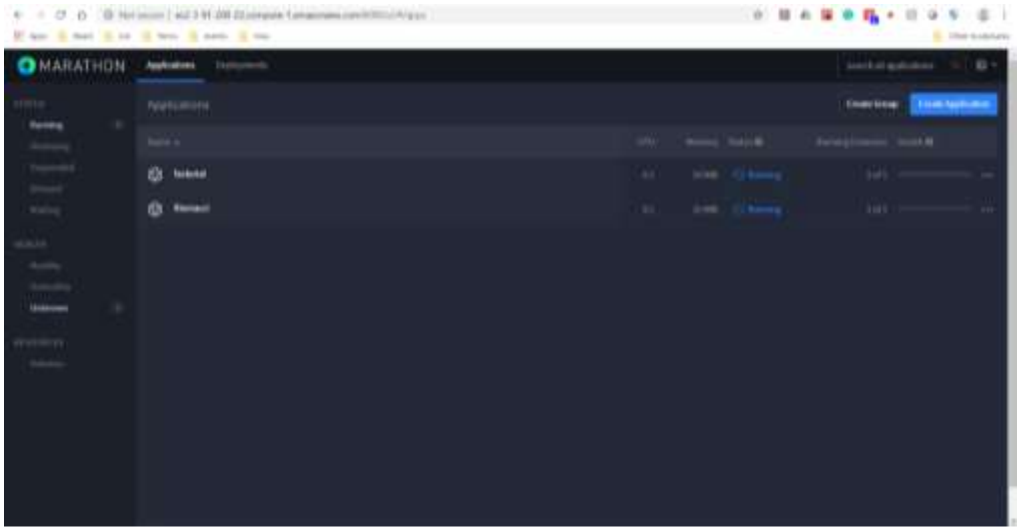


Figure 15Marathon UI showing factorial and Fibonacci application

Configuring Fibonacci and factorial program

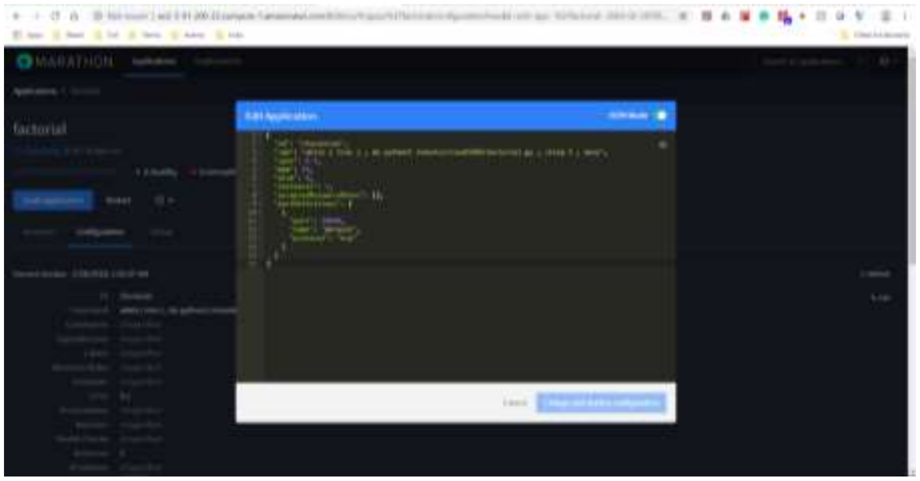


Figure 16 Configuring the factorial program

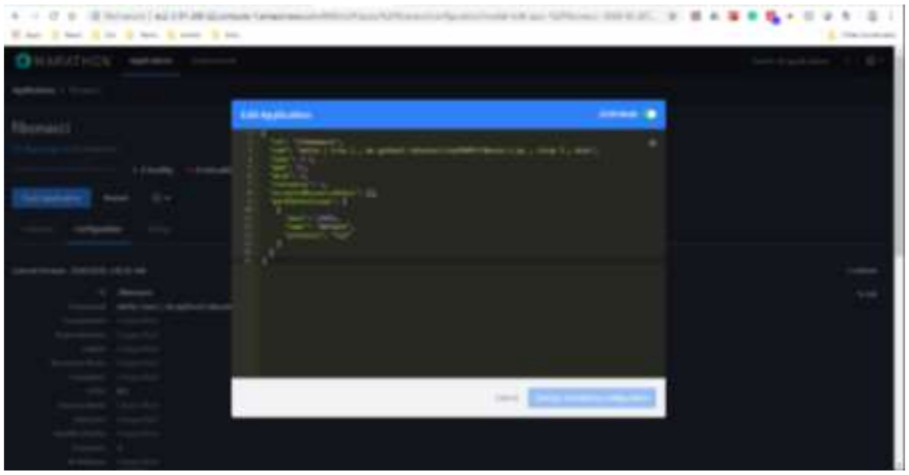


Figure 17 Configuring Fibonacci program

Applications that are created on marathon are registered with Mesos and are visible in active tasks

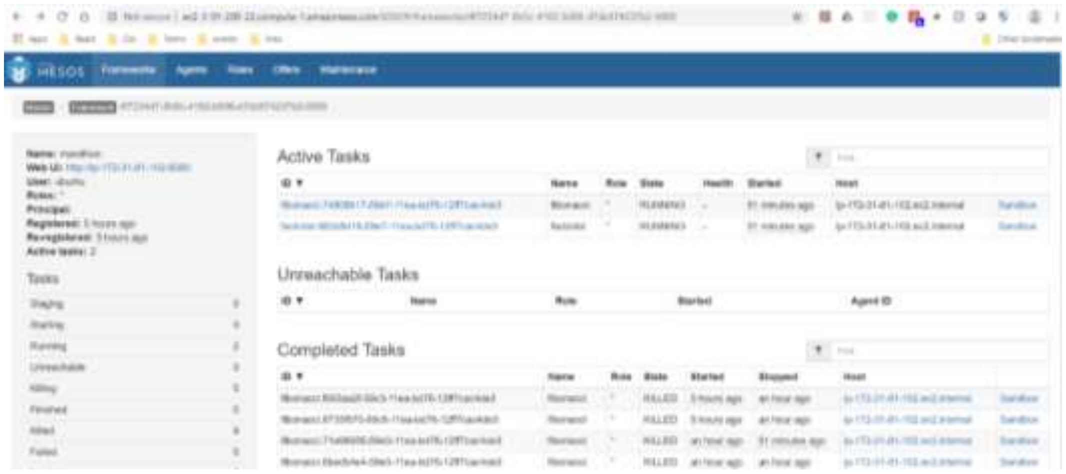


Figure 18 Active Tasks in Mesos

Graphs generated for the factorial and Fibonacci for each N value

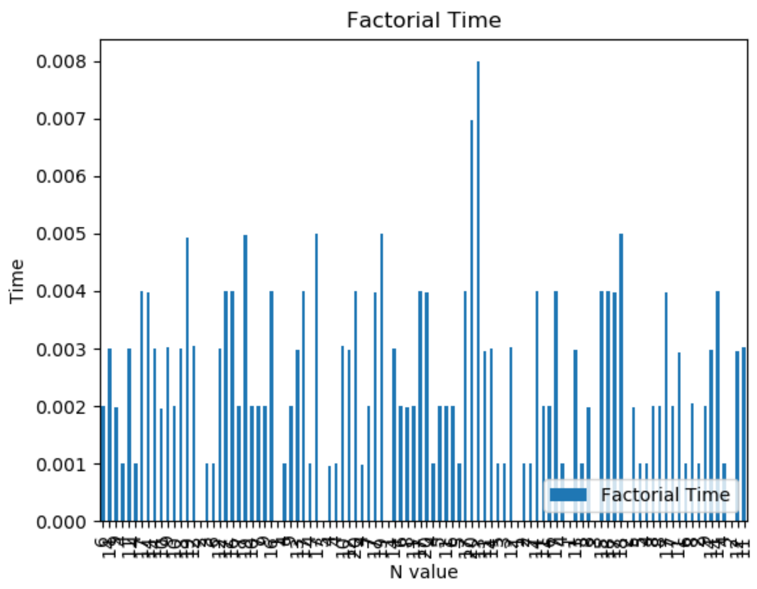


Figure 19 Factorial Time Graph

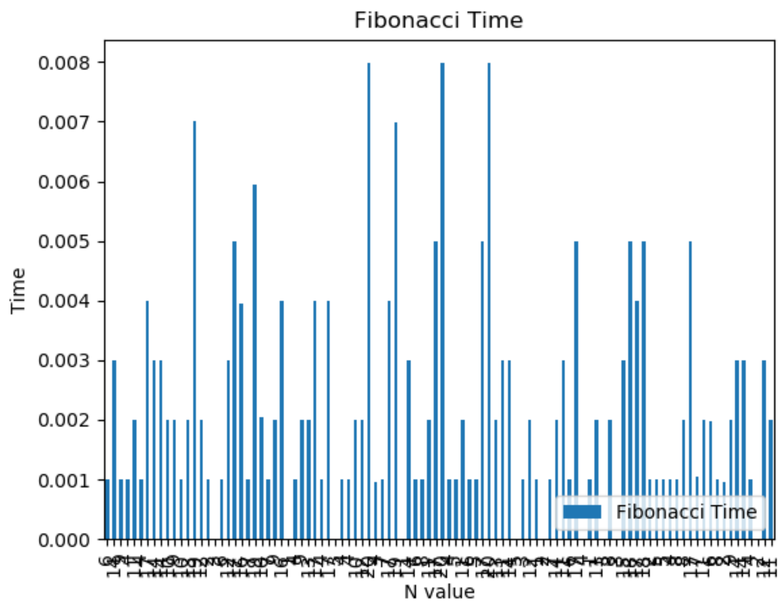


Figure 20 Fibonacci Time Graph

References

[1]"How to Manually Install Java 8 on Ubuntu 16.04", *Vultr*, 2020. [Online]. Available: https://www.vultr.com/docs/how-to-manually-install-java-8-on-ubuntu-16-04. [Accessed: 27- Feb- 2020].

[2]"How-To Guide – Introduction to Apache Mesos – Linux Academy", *Linux Academy*, 2020. [Online]. Available: https://linuxacademy.com/guide/25034-introduction-to-apache-mesos/. [Accessed: 27- Feb- 2020].

[3] “IEEE Xplore Full-Text PDF:,” *Library.dal.ca*, 2020. [Online]. Available: https://ieeexploreieee-org.ezproxy.library.dal.ca/stamp/stamp.jsp?tp=&arnumber=8457829&tag=1. [Accessed: 24-Feb-2020]

[4]"Marathon: Install Marathon", *Mesosphere.github.io*, 2020. [Online]. Available: https://mesosphere.github.io/marathon/docs/. [Accessed: 27- Feb- 2020].

[5]"How-To Guide – Container Orchestration using Mesos and Marathon – Linux Academy", *Linux Academy*, 2020. [Online]. Available: https://linuxacademy.com/guide/28039-container-orchestration-using-mesos-and-marathon/. [Accessed: 27- Feb- 2020].

[6]"Apache Mesos", *Apache Mesos*, 2020. [Online]. Available: http://mesos.apache.org/documentation/latest/. [Accessed: 27- Feb- 2020].

[7]"Marathon: A container orchestration platform for Mesos and DC/OS", *Mesosphere.github.io*, 2020. [Online]. Available: https://mesosphere.github.io/marathon/. [Accessed: 27- Feb- 2020].