

PANIMALAR ENGINEERING COLLEGE

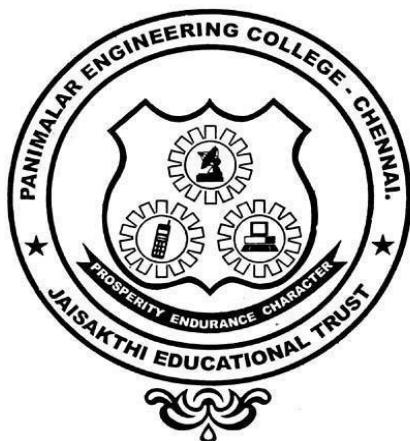
(A CHRISTIAN MINORITY INSTITUTION)

JAISAKTHI EDUCATIONAL TRUST

ACCREDITED BY NATIONAL BOARD OF ACCREDITATION (NBA)

Bangalore Trunk Road, Varadharajapuram, Nasarathpettai,

Poonamallee, Chennai – 600 123.



RECORD NOTE BOOK

STUDENT NAME: ARJUN M

DEPARTMENT : B.TECH. INFORMATION TECHNOLOGY

REGISTER NUMBER: 211419205015 SEMESTERS: SEVENTH

SUBJECT CODE & NAME: IT8711 – FOSS AND CLOUD COMPUTING LAB.

ACADEMIC YEAR: 2022 - 2023

TABLE OF CONTENTS

| Sl.No. | Date | Name Of the Experiment | Page No. | Marks | Signature |
|--------|------------|---|----------|-------|-----------|
| 1. a. | 22-08-2022 | Use gcc to compile c-programs. Split the programs to different modules and create an Grade application using make command. | | | |
| b. | 22-08-2022 | Use gcc to compile c-programs. Split the programs to different modules and find the greatest of two numbers | | | |
| 2. | 29-08-2022 | Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories | | | |
| 3. a. | 5-09-2022 | Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8. | | | |
| b. | 5-09-2022 | Find Procedure to Run the Virtual Machine of Different Configuration. Check How Many Virtual Machines Can Be Utilized at Particular Time. | | | |
| 4. a. | 19-09-2022 | Install a C compiler in the virtual machine created using virtual box and execute Arithmetic Operation Programs | | | |
| 4. b. | 19-09-2022 | Install a C++ compiler in the virtual machine created using virtual box and find sum of the digits of a given number | | | |
| 5. | 10-10-2022 | Install Google App Engine. Create hello world web applications using python/java. | | | |
| 6. | 10-10-2022 | Use GAE launcher to launch the 'Hello world! 'Web applications. | | | |
| 7. | 17-10-2022 | Find a procedure to transfer the files from one virtual machine to another virtual machine. | | | |
| 8. | 17-10-2022 | Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version) | | | |
| 9. a. | 07-11-2022 | Find procedure to set up the one node hadoop cluster | | | |
| b. | 07-11-2022 | Write A Wordcount Program to Demonstrate the Use of Map And Reduce Tasks | | | |
| 10. | 07-11-2022 | Simulate a cloud scenario using CloudSim and run a FCFS scheduling algorithm that is not present in CloudSim. | | | |

AVERAGE:

PANIMALAR ENGINEERING COLLEGE

**(A CHRISTIAN MINORITY INSTITUTION)
JAISAKTHI EDUCATIONAL TRUST**

**ACCREDITED BY BOARD OF ACCREDITATION – NBA, NEW DELHI
Bangalore Trunk Road, Varadharajapuram, Nasarathpettai,
Poonamallee (TK) Chennai – 600 123**

DEPARTMENT OF INFORMATION TECHNOLOGY



BONAFIDE CERTIFICATE

This is a Certified bonafide Record Book of Mr./Ms.ARJUN M.....

**Register Number ...211419205015... Submitted for Anna University Practical
Examination held on _____ in IT8711 – FOSS AND CLOUD
COMPUTING LABORATORY during December 2022.**

Staff - In charge

External Examiner

Internal Examiner

Aim :

To create an application by split the programs to different modules using make command and gcc to compile c-programs .

Introduction :

Module : Separate a set of code into a self-contained entity.

A module is also referred as called a library , code that is in an already-compiled form.

Make File :

The make utility automatically determines which pieces of a large program need to be recompiled, and issues commands to recompile them.

You need a file called a *makefile* to tell make what to do. The makefile tells make how to compile and link a program.

Save in the same directory as the source file.

Use "tab" to indent the command (NOT spaces).

Syntax :

target: pre-requisites -1 pre-requisites -2 ...

command

Note :

- ❖ The *target* and *pre-requisites* are separated by a colon (:). The *command* must be preceded by a tab (NOT spaces).
- ❖ A *target* is usually the name of a file that is generated by a program.
- ❖ A target can also be the name of an action to carry out
- ❖ A *prerequisite* is a file that is used as input to create the target.
- ❖ *command* is an action that make carries out.
- ❖ **you need to put a tab character at the beginning of every *command* line!**

Automatic Variables

Automatic variables are set by make after a rule is matched. There include:

- \$@: the target filename.

- \$*: the target filename without the file extension.
- \$<: the first prerequisite filename.
- \$^: the filenames of all the prerequisites, separated by spaces, discard duplicates.
- \$+: similar to \$^, but includes duplicates.
- \$?: the names of all prerequisites that are newer than the target, separated by spaces.

A target that does not represent a file is called a **phony target**.

You can use **VPATH (uppercase)** to specify the directory to search for dependencies and target files.

Procedure :

To create a C module for the grade functions

1. We need to create 2 new source code files: **grades.h** file and **grades.c** file
2. **Grades.h** file contain details about the *interface* (*i.e. Functional prototypes*) to the module.
3. **Grades.c** file contain details about the *implementation* of the module which is already declared in .h file
4. **Adjust.c** file consist of main program and *user* of the grades module
5. The PrintGrades(float grades[], int howmany) module is used for Grades are all printed on one line separated by spaces and terminated by a newline.
6. The AdjustGrades(float grades[], float adjustments[], int howmany) module is Adjusts each grade in the `grades' array using the corresponding change in the `adjustments' array.
7. The AdjustedGrade(float grade, float adjustment) module is Adjusts one grade and returns the new grade.
8. Create the Make file named as “Makefile “ to run the grade application in c program.
9. Draw the dependency chart for know exactly what commands are needed to generate the pieces that make up the executable.
10. Stop the process

Program :

grades235.h

```
***** Function Prototypes *****
```

```
void PrintGrades(float grades[], int howmany);
void AdjustGrades(float grades[], float adjustments[], int howmany);
float AdjustedGrade(float grade, float adjustment);
```

grades235.c

```
#include <stdio.h>
```

```

#include "grades235.h"

void PrintGrade(float grades[], int howmany)
{
    int i;
    for (i = 0; i < howmany; i++)
    {
        printf(" %.2f", grades[i]);
    }
    printf("\n");
}

void AdjustGrades(float grades[], float adjustments[], int howmany)
{
    float newGrade;
    int i;
    for (i = 0; i < howmany; i++)
    {
        /* This two-step assignment can be done in one step. */
        newGrade = AdjustedGrade(grades[i], adjustments[i]);
        grades[i] = newGrade;
    }
}

float AdjustedGrade(float grade, float adjustment)
{
    grade += adjustment;
    return grade;
}

```

adjust235.c

```

#include <stdio.h>
#include "grades235.h"
#define MAX_GRADES 10
int main()
{

```

```

float adjustments[MAX_GRADES];      /* Grade adjustments for all students. */
float grades[MAX_GRADES];          /* Grades for current student.      */
int numGrades;                     /* Number of adjustments.         */
int numStudents;                   /* Number of students.           */
int studentNum;                   /* Index of current student.     */
int i;
printf("How many grades per student? ");
scanf("%d", &numGrades);
if (numGrades< 1 || numGrades> MAX_GRADES) {
    printf("I can only handle ..%d grades!", MAX_GRADES);
    exit(1); /* Terminate program, return error status. */
}
for (i = 0; i <numGrades; i++) {
    printf("\nEnter adjustment for %d> ", i+1);
    scanf("%f", &adjustments[i]);
}
printf("\nHow many students? ");
scanf("%d", &numStudents);
for (studentNum = 1; studentNum<= numStudents; studentNum++) {
    printf("\n");
    /* Get each grade for the current student. */
    printf("Student #%-d\n", studentNum);
    for (i = 0; i <numGrades; i++) {
        printf("\nEnter grade for %d : ", i+1);
        scanf("%f", &grades[i]);
    }
    /* Print grades before and after adjustment. */
    printf("\n\nOld Student #%-d grades:", studentNum);
    PrintGrades(grades, numGrades);
    AdjustGrades(grades, adjustments, numGrades);
    printf("\nNew Student #%-d grades:", studentNum);
    PrintGrades(grades, numGrades);
}

```

```
return 0;  
}
```

Makefile

```
# Makefile for executable adjust
```

```
# Parameters to control Makefile operation
```

```
CC = gcc
```

```
CFLAGS = -ansi -pedantic -Wall -g
```

```
# ****
```

```
# Entries to bring the executable up to date
```

```
adjust: adjust.ogrades.o
```

```
    $(CC) $(CFLAGS) -o adjust adjust.ogrades.o
```

```
adjust.o: adjust.cgrades.h
```

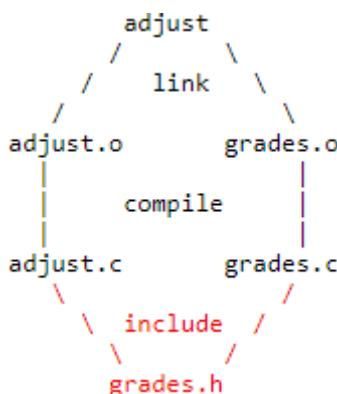
```
    $(CC) $(CFLAGS) -c adjust.c
```

```
grades.o: grades.cgrades.h
```

```
    $(CC) $(CFLAGS) -c grades.c
```

Dependency Chart :

- ❖ The executable **adjust235** is made up of the 2 object files, **adjust235.o** and **grades235.o**. Thus, **adjust** depends on those 2 files. To generate **adjust** from those files, we *link* them together.
- ❖ Next, **adjust235.o** depends on **adjust235.c** and **grades235.o** depends on **grades235.c**.
- ❖ Finally, the 2 source code files depend on **grades235.h** since they both *include* that file.



Sample Input and Output :

```
How many grades per student? 5
```

```
Enter adjustment for 1> 1
```

```
Enter adjustment for 2> 2
```

Enter adjustment for 3> 3

Enter adjustment for 4> 2

Enter adjustment for 5> 1

How many students? 2

Student #1

Enter grade for 1 : 67

Enter grade for 2 : 85

Enter grade for 3 : 95

Enter grade for 4 : 65

Enter grade for 5 : 79

Old Student #1 grades: 67.00 85.00 95.00 65.00 79.00

New Student #1 grades: 68.00 87.00 98.00 67.00 80.00

Student #2

Enter grade for 1 : 95

Enter grade for 2 : 84

Enter grade for 3 : 85

Enter grade for 4 : 95

Enter grade for 5 : 69

Old Student #2 grades: 95.00 84.00 85.00 95.00 69.00

New Student #2 grades: 96.00 86.00 88.00 97.00 70.00

Output:

```
 sam@sam-ThinkCentre-M72e: ~/arj235/exp1
 sam@sam-ThinkCentre-M72e:~/arj235/exp1$ gcc adjust.235.c
 sam@sam-ThinkCentre-M72e:~/arj235/exp1$ ./a.out
 How many grades per student? 2
 Enter adjustment for 1> 10
 Enter adjustment for 2> 20
 >_ How many students? 3
 Student #1
 Enter grade for 1 :  50
 Enter grade for 2 :  55
 Old Student #1 grades: 50.00 55.00
 New Student #1 grades: 60.00 75.00
 Student #2
 Enter grade for 1 :  60
 Enter grade for 2 :  55
 Old Student #2 grades: 60.00 55.00
 New Student #2 grades: 70.00 75.00
 Student #3
 Enter grade for 1 :  60
 Enter grade for 2 :  70
 Old Student #3 grades: 60.00 70.00
 New Student #3 grades: 70.00 90.00
 sam@sam-ThinkCentre-M72e:~/arj235/exp1$
```

Result :

The development of grade application by split the programs to different modules using make command and gcc to compile c-programs was successfully written ,executed and verified.

Aim :

To create an application by split the programs to different modules using make command and gcc to compile c-programs .

Introduction :

Module : Separate a set of code into a self-contained entity. A module is also referred as called a library , code that is in an already-compiled form.

Make File :

The make utility automatically determines which pieces of a large program need to be recompiled, and issues commands to recompile them. You need a file called a makefile to tell make what to do. The makefile tells make how to compile and link a program. Save in the same directory as the source file. Use "tab" to indent the command (NOT spaces).

Syntax :

Note :

- ❖ The target and pre-requisites are separated by a colon (:). The command must be preceded by a tab (NOT spaces).
- ❖ A target is usually the name of a file that is generated by a program. A target can also be the name of an action to carry out A prerequisite is a file that is used as input to create the target.
- ❖ command is an action that make carries out.
- ❖ you need to put a tab character at the beginning of every command line!

Automatic Variables:

Automatic variables are set by make after a rule is matched.

- There include: \$@: the target filename.
- \$*: the target filename without the file extension.
- \$<: the first prerequisite filename.
- \$^: the filenames of all the prerequisites, separated by spaces, discard duplicates.
- \$?: the names of all prerequisites that are newer than the target, separated by spaces.

- A target that does not represent a file is called a phony target. You can use VPATH (uppercase) to specify the directory to search for dependencies and target files.

Procedure :

1. We need to create 2 new source code files: greatest235.h file and greatest 235.c file
2. greatest 235.h file contain details about the interface (i.e.Functional prototypes) to the module.
3. greatest 235.c file contain details about the implementation of the module which is already declared in .h file
4. adjust235.c file consist of main program and user of the grades module
5. The void Greatest(int num1, int num2); module is used for greatest are all printed on one line separated by spaces and terminated by a newline.
6. The Greatest(int num1, int num2) module is adjusts the greatest number array using the corresponding change in the `adjustments' array.
7. The adjust235 module is finds the greatest of number and returns it.
8. Create the Make file named as “Makefile “ to run the grade application in c program.
9. Draw the dependency chart for know exactly what commands are needed to generate the pieces that make up the executable.
10. Stop the process.

Program :

greatest235.h

```
***** Function Prototypes *****
```

```
void Greatest(int num1, int num2);
```

greatest235.c

```
#include<stdio.h>
#include"greatest259.h"
void Greatest(int num1, int num2)
{
if(num1>num2)
{
printf("Number1 is bigger");
}
else
{
printf("Number2 is bigger");
}
}
```

adjust235.c

```
#include<stdio.h>
#include<stdlib.h>
#include"greatest259.c"

int main()
{
int num1,num2;
printf("Enter Number1\n");
scanf("%d", &num1);
printf("Enter Number2\n");
scanf("%d", &num2);
Greatest(num1,num2);
return 0;
}
```

Makefile

```
# Makefile for executable adjust
```

```
# Parameters to control Makefile operation CC
= gcc,
CFLAGS = -ansi -pedantic -Wall -g

# ****
# Entries to bring the executable up to date

adjust: adjust235.ogrades235.o

$(CC) $(CFLAGS) -o adjust adjust235.ogrades235.o

adjust235.o: adjust.cgrades.h

$(CC) $(CFLAGS) -c adjust.c
```

grades235.o: grades.cgrades.h

\$(CC) \$(CFLAGS) -c grades.c

Dependency Chart :

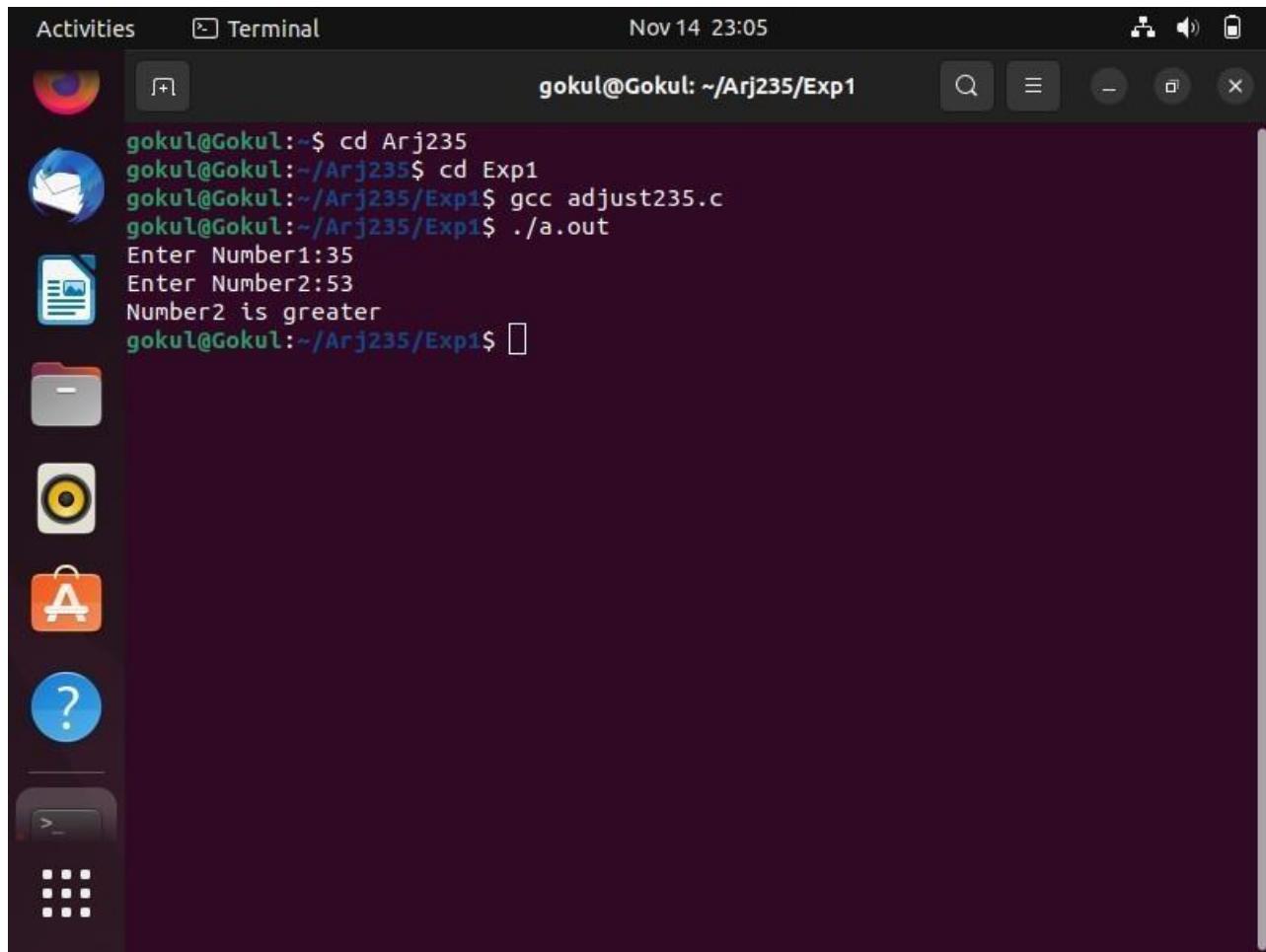
The executable **adjust** is made up of the 2 object files, **adjust235.o** and **grades235.o**. Thus, **adjust** *depends* on those 2 files. To generate **adjust** from those files, we *link* them together.

Next, **adjust235.o** depends on **adjust235.c** and **greatest235.o** depends on **greatest235.c**. Finally, the 2 source code files depend on **greatest235.h** since they both *include* that file.

Sample Input and Output :

Enter the 1st number

Enter the 2nd number



A screenshot of an Ubuntu desktop environment. On the left, there's a vertical dock with icons for Dash, Home, Applications, Help, and a terminal window. The terminal window is active and shows the following command-line session:

```
gokul@Gokul:~$ cd Arj235
gokul@Gokul:~/Arj235$ cd Exp1
gokul@Gokul:~/Arj235/Exp1$ gcc adjust235.c
gokul@Gokul:~/Arj235/Exp1$ ./a.out
Enter Number1:35
Enter Number2:53
Number2 is greater
gokul@Gokul:~/Arj235/Exp1$
```

Result :

The development of grade application by split the programs to different modules using make command and gcc to compile c-programs was successfully written ,executed and verified.

USE VERSION CONTROL SYSTEMS COMMAND TO CLONE, COMMIT, PUSH, FETCH, PULL, CHECKOUT, RESET, AND DELETE REPOSITORIES.

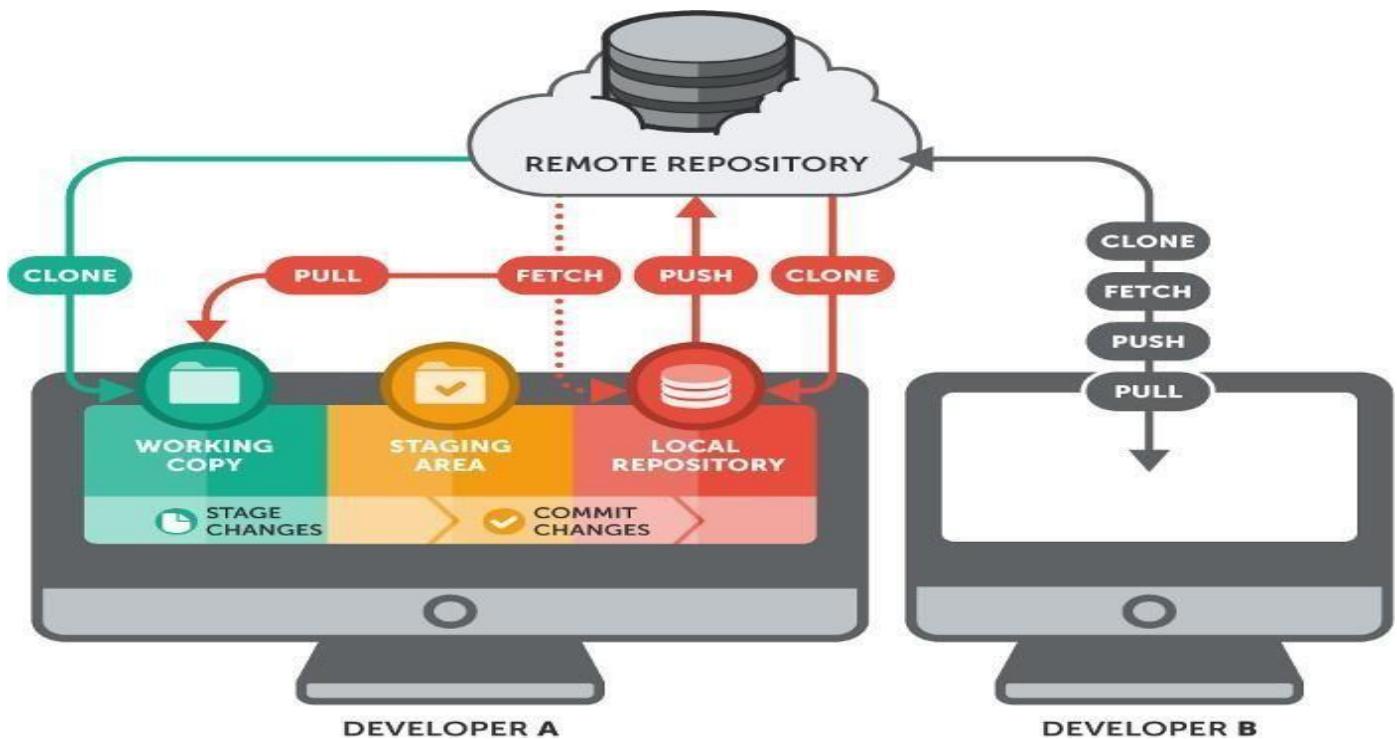
Aim :

To use the version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories.

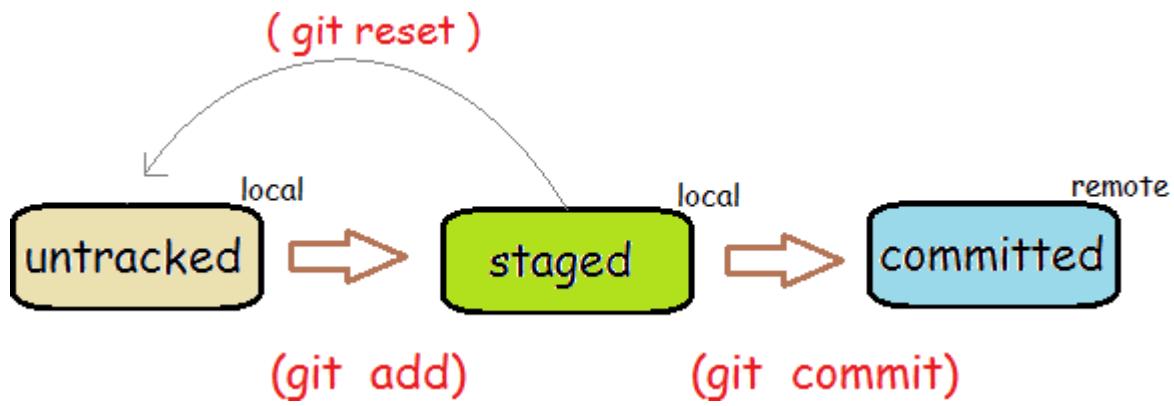
Procedure :

Introduction :

- A **repository** is a place where you will have your codes or you can imagine a foreground server.
- Git is a fast distributed revision control system.
- **Git** is a *version control system (software)* and **GitHub** is a *source code hosting service*. Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people
- Git is best thought of as a tool for storing the history of a collection of files. It stores the history as a compressed collection of interrelated snapshots of the project's contents. In Git each such version is called a commit.



A file in git goes through the following stages:



Your local repository consists of three "**trees**" maintained by git.

- 1) The first one is your **Working Directory** which holds the actual files.
- 2) The second one is the **Index** which acts as a **staging area** and
- 3) Finally the **HEAD** which points to **the last commit** you've made.

The Working Tree is the area where you are currently working. It is where your files live. This area is also known as the "**untracked**" area of git.

The **StagingArea** is when git starts tracking and saving changes that occur in files. The saved changes reflect in the .git directory. That is about it when it comes to the Staging Area.

- How can you see what is in your Working Tree?
- Run the command **git status**. This command will show you **two things**:
 - **The files in your Working Tree** and
 - **the files in your Staging Area**.
- Work may simultaneously proceed along parallel lines of development, called **branches**, which may merge.
- A single Git repository **can track development on multiple branches**. It does this by keeping a list of **heads** which reference the latest commit on each branch
 - **\$ git branch**
 - *** master**

Branch :

- "branch" **to mean a line of development**, and "branch head" (or just "head") **to mean a reference to the most recent commit on a branch**

Manipulating branches:

- **git branch**
 - list all branches.
- **git branch <branch>**
 - create a new branch named <branch>, referencing the same point in history as the current branch.
- **git branch <branch><start-point>**
 - create a new branch named <branch>, referencing <start-point>, which may be specified any way you like, including using a branch name or a tag name.
- **git branch -d <branch>**
 - delete the branch <branch>; if the branch is not fully merged in its upstream branch or contained in the current branch, this command will fail with a warning.
- **git branch -D <branch>**
 - delete the branch <branch> irrespective of its merged status.
- **git switch <branch>**
 - make the current branch <branch>, updating the working directory to reflect the version referenced by <branch>.
- **git switch -c <new><start-point>**
 - create a new branch <new> referencing <start-point>, and check it out.
- The special symbol "HEAD" can **always be used to refer to the current branch**. In fact, Git uses a file named HEAD in the .git directory to remember which branch is current:
- **\$ cat .git/HEAD**

ref: refs/heads/master
- **Examining branches from a remote repository**
- **\$ git branch -r**
 - **origin/HEAD**
 - **origin/html**
 - **origin/maint**
 - **origin/man**
 - **origin/master**

- **origin/next**
- **origin/seen**
- **origin/todo**

```
$ git tag -l
```

```
v2.6.11  
v2.6.11-tree  
v2.6.12  
v2.6.12-rc2  
v2.6.12-rc3  
v2.6.12-rc4  
v2.6.12-rc5  
v2.6.12-rc6  
v2.6.13
```

- Tags are expected to **always point at the same version of a project**, while **heads** are expected to **advance as development progresses**
- So, First, you need to run **git init** command, because it will create a new git repository and then after you can perform **git status** *the* command to check to your in Working Tree.
- You can see that the **untracked files** are shown **in red color**. If you don't run the gitinit command you will get an error, which looks like something in the image below.

```
HOME@LAPTOP-MSQ6RBU0 MINGW64 /g/git-tutorial  
$ git init  
Initialized empty Git repository in G:/git-tutorial/.git/  
  
HOME@LAPTOP-MSQ6RBU0 MINGW64 /g/git-tutorial (master)  
$ git status  
On branch master  
  
No commits yet  
  
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
  
    about.txt.txt  
    index.txt.txt  
  
nothing added to commit but untracked files present (use "git add" to track)
```

- Running the command ***git add [filename]*** will add a specific file to the Staging Area from your Working Tree. If you want to add everything from the Working Tree, then run the command ***git add .***
- The dot (.) operator is a wildcard meaning all files

```
HOME@LAPTOP-MSQ6RBU0 MINGW64 /g/git-tutorial (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    about.txt.txt
    index.txt.txt

nothing added to commit but untracked files present (use "git add" to track)

HOME@LAPTOP-MSQ6RBU0 MINGW64 /g/git-tutorial (master)
$ git add .

HOME@LAPTOP-MSQ6RBU0 MINGW64 /g/git-tutorial (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   about.txt.txt
    new file:   index.txt.txt

HOME@LAPTOP-MSQ6RBU0 MINGW64 /g/git-tutorial (master)
$ |
```

The **clone** command :

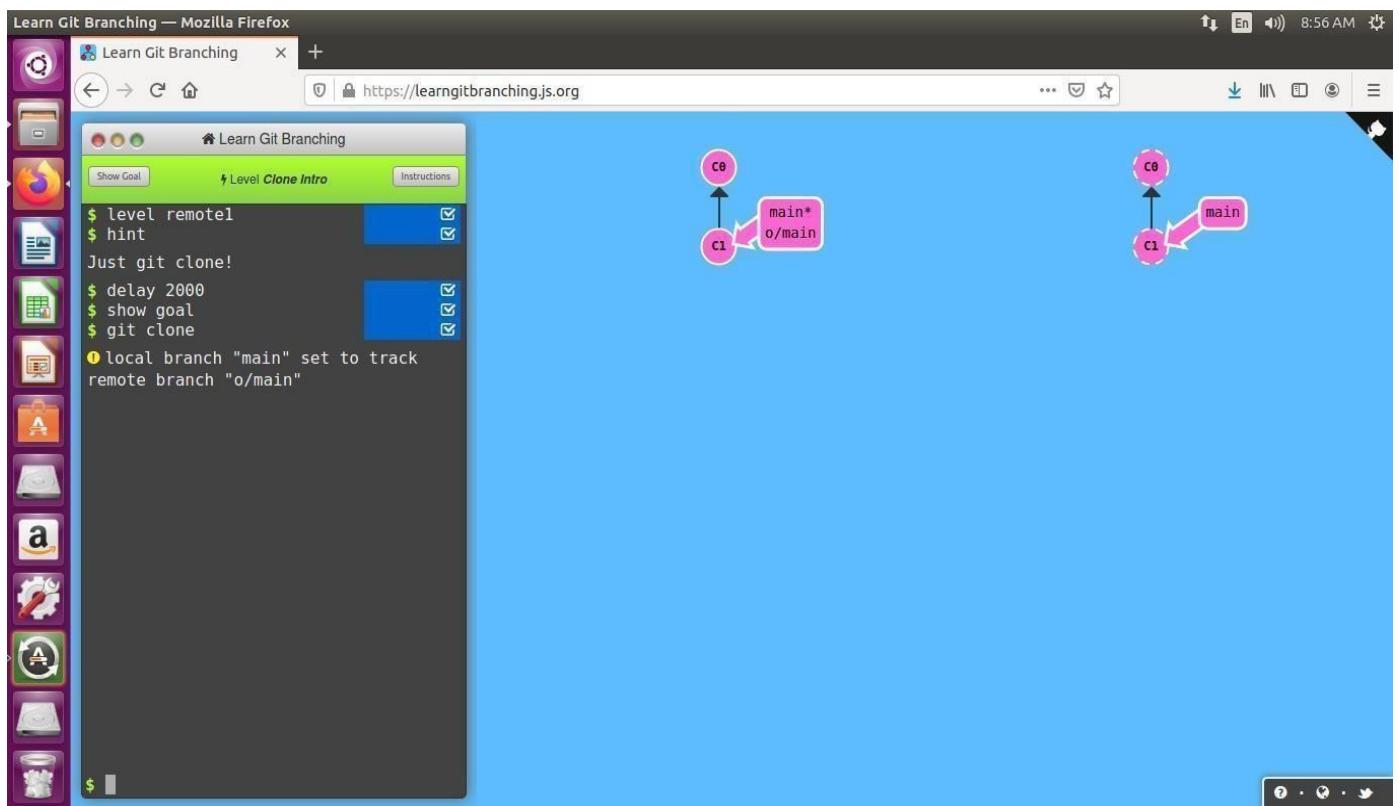
The **clone** command creates a new directory named after the project (git or linux in the examples above). After you cd into this directory, you will see that it contains a copy of the project files, called the working tree, together with a special top-level directory named .git

Syntax :**git clone [URL]**

- # Git itself (approx. 40MB download):
 - **\$ git clone git://git.kernel.org/pub/scm/git/git.git**
- # the Linux kernel (approx. 640MB download):
 - **\$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git**

A new local repository is automatically created, with the github version configured as a remote.

The initial clone may be **time-consuming for a large project**, but you will **only need to clone once**.

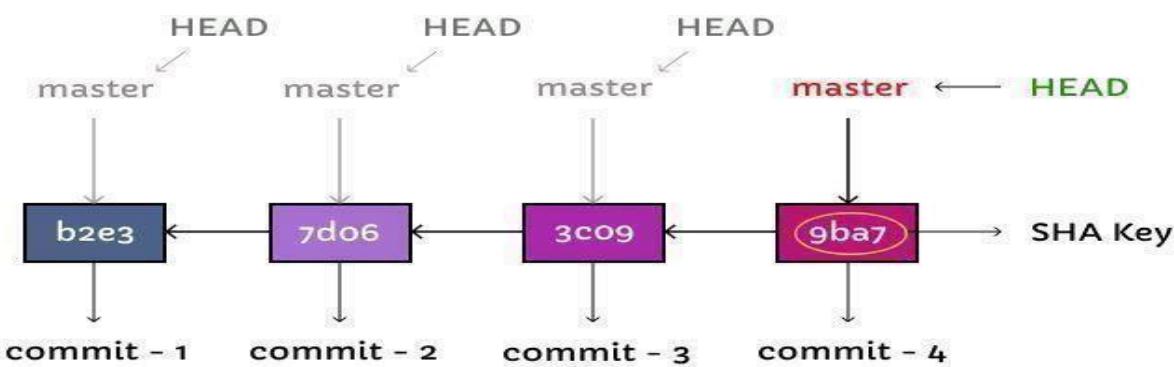


The git commit command :

A commit in a git repository records a snapshot of all the (tracked) files in your directory. It's like a giant copy and paste, but even better.

How is git commit command works internally?

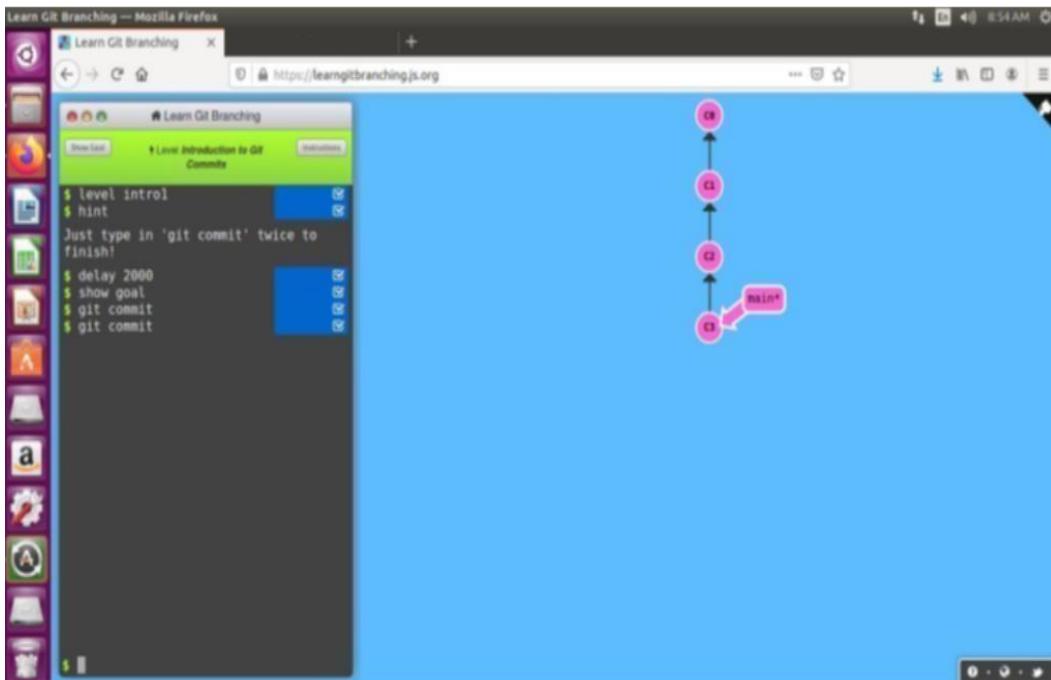
The above image shows the working of git commit command. In which the four blocks represent the commit with the unique **SHA** commit key. Suppose, you are on the **master** branch and you did a commit now git internally will create a block (node) of a commit you did. In git, we have a pointer called **HEAD** (It is the reference to the commit in the current branch in our case i.e master). When you commit something the HEAD will be pointed to our commit in our case the commit hash key is **b2e3** . If you made another commit now HEAD will move and point to commit having a hash key **7d06** means the parent of the **7d06** commit



is **b2e3** because it holds the reference of the previous commit. As you see our last commit is **9ba7** and the parent of this commit is **3c09** and our HEAD is pointing to **9ba7** commit.

Syntax :git commit -m “[Type in the commit message]”

\$ git commit -m “First Commit”



- Every change in the history of a project is represented by a commit.
- **A commit shows who made the latest change, what they did, and why.**
- The git-show command shows the ***most recent commit on the current branch***:

– **\$ git show**

```
commit 17cf781661e6d38f737f15f53ab552f1e95960d7
```

Author: Abcchand

Date: Tue Nov 3 14:11:06 2020 -0700

Remove duplicate getenv(DB_ENVIRONMENT) call

Noted by Tony Luck.

```
diff --git a/init-db.c b/init-db.c
```

```
index 65898fa..b002dc6 100644
```

```
--- a/init-db.c
```

```
+++ b/init-db.c
```

```
@@@ -7,7 +7,7 @@@
```

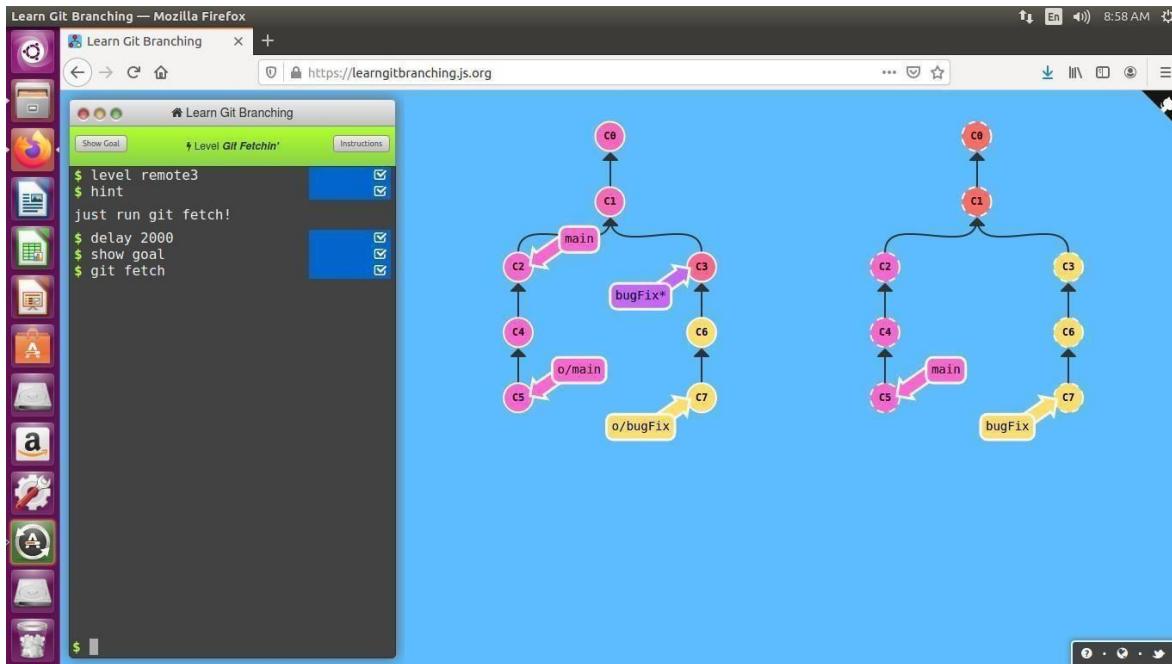
The git fetch command :

- The gitfetch command, with no arguments, will update all of the remote-tracking branches to the latest version found in the original repository.

Update the remote-tracking branches:

```
$ git fetch origin
```

When we run this command, a copy of our remote repository (located at “origin”) is downloaded and saved to our local machine. However, the local copy of our code has not yet been changed.



Uploading to a server - git push :

To transfer our local commits to the server. This process is called a **push**, and is done every time we want to update the remote repository.

The Git command to do this is git push and **takes two parameters** - the **name of the remote repo** (we called ours *origin*) and the **branch to push to** (*master* is the default branch for every repo).

```
$ git push origin master
```

```

Counting objects: 3, done.
Writing objects: 100% (3/3), 212 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/tutorialzine/awesome-project.git
 * [new branch]    master -> master

```

Learn Git Branching — Mozilla Firefox

Learn Git Branching

Show Goal ↗ Level Git Pushin' Instructions

```
$ level remote6
$ hint
Remember you have to clone before you can push!
$ delay 2000
$ show goal
$ git clone
An origin already exists! You cannot make a new one
$ git commit
$ git push
$ git commit
$ git push
```

Getting changes from a server – git pull :

If you make updates to your repository, people can download your changes with a single command
- **pull:**

\$ git pull origin master

From <https://github.com/tutorialzine/awesome-project>

* branch master -> FETCH_HEAD

Already up-to-date.

Learn GIT Branching — Mozilla Firefox

Learn Git Branching

Show Goal ↗ Level Git Pullin' Instructions

```
$ level remote4
$ hint
Just run git pull!
$ delay 2000
$ show goal
$ git pull
```

Creating new branches - gitbranch :

The default branch of every repository is called **master**. To create additional branches use the git branch <name> command:

```
$ git branch FinalIT
```

This just creates the new branch, which at this point is exactly the same as our *master*.

Switching branches - gitcheckout :

Now, when we run git branch, we will see there are two options available:

```
$ git branch
```

```
FinalIT
```

```
* master
```

Master is the current branch and is marked with an asterisk. However, we want to work on our new FinalIT, so we need to switch to the other branch. This is done with the git checkout command, expecting one parameter - the branch to switch to.

```
$ git checkout FinalIT
```

```
$ git branch
```

```
master
```

```
* FinalIT
```

Switched to branch 'FinalIT'

The git reset command :

This command undoes all the commits after the specified commit and preserves the changes locally.

The git reset command allows you to RESET your current head to a specified state. You can reset the state of specific files as well as an entire branch.

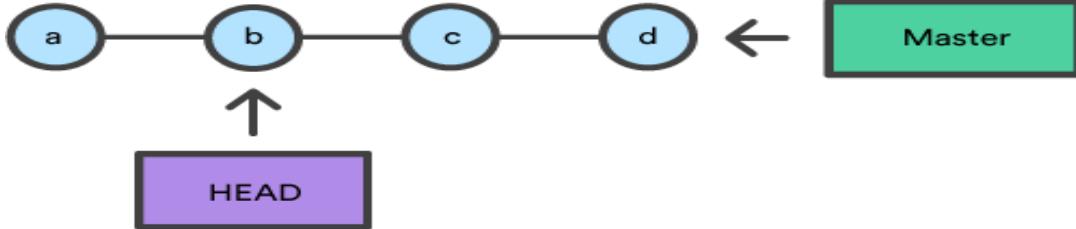
Git reset is similar in behavior to git checkout. Where git checkout solely operates on the HEAD ref pointer, **git reset will move the HEAD ref pointer and the current branch ref pointer**. To better demonstrate this behavior consider the following example:

This example demonstrates a sequence of commits on the master branch. The HEAD ref and master branch ref currently point to commit d. Now let us execute and compare, both

```
$ git checkout b
```

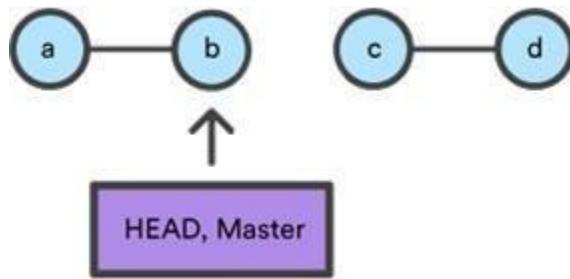
```
$ git reset b.
```

\$git checkout b



With git checkout, the master ref is still pointing to d. The HEAD ref has been moved, and now points at commit b. The repo is now in a 'detached HEAD' state.

\$git reset b



Comparatively, git reset, moves both the HEAD and branch refs to the specified commit.

Git rm :

This command deletes the file from your working directory and stages the deletion.

Syntax :git rm [file]

Example :

```
$ git rm sample.txt
```

```
$ git branch -d finalit
```

This command deletes the finalit branch

Result :

The useof version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories was learned and verified successfully.

INSTALL VIRTUALBOX/VMWARE WORKSTATION WITH DIFFERENT FLAVOURS OF LINUX OR WINDOWS OS ON TOP OF WINDOWS7 OR 8.

Aim :

To find and learn the procedure to Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.

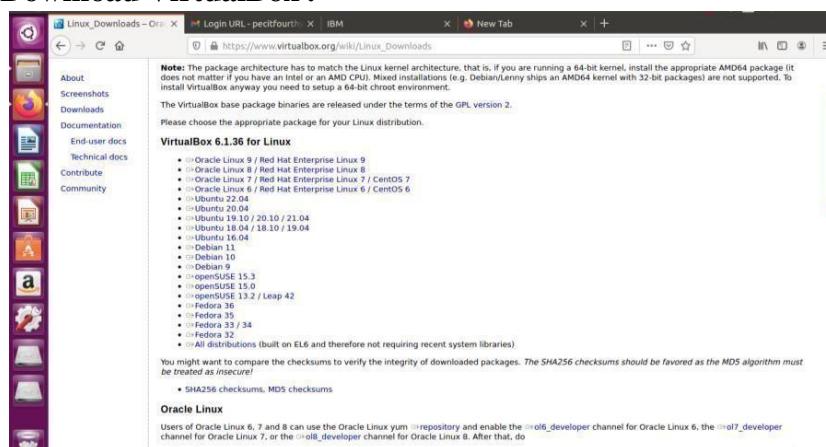
Procedure :

Install Virtualbox :

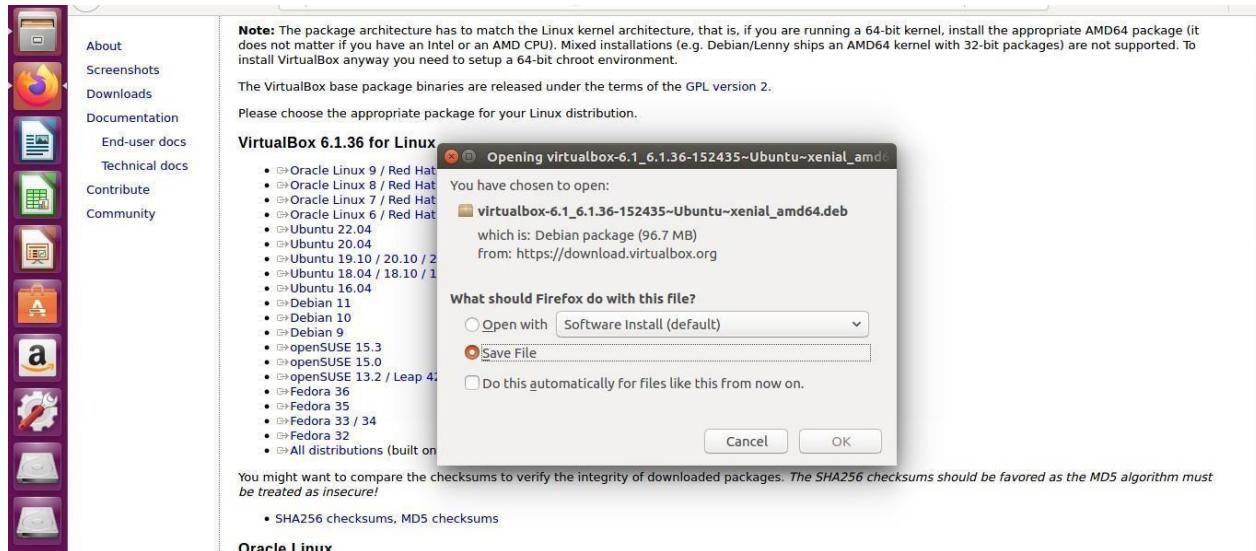
1. **Open the VirtualBox website.** Go to <https://www.virtualbox.org/> in your computer's Internet browser.



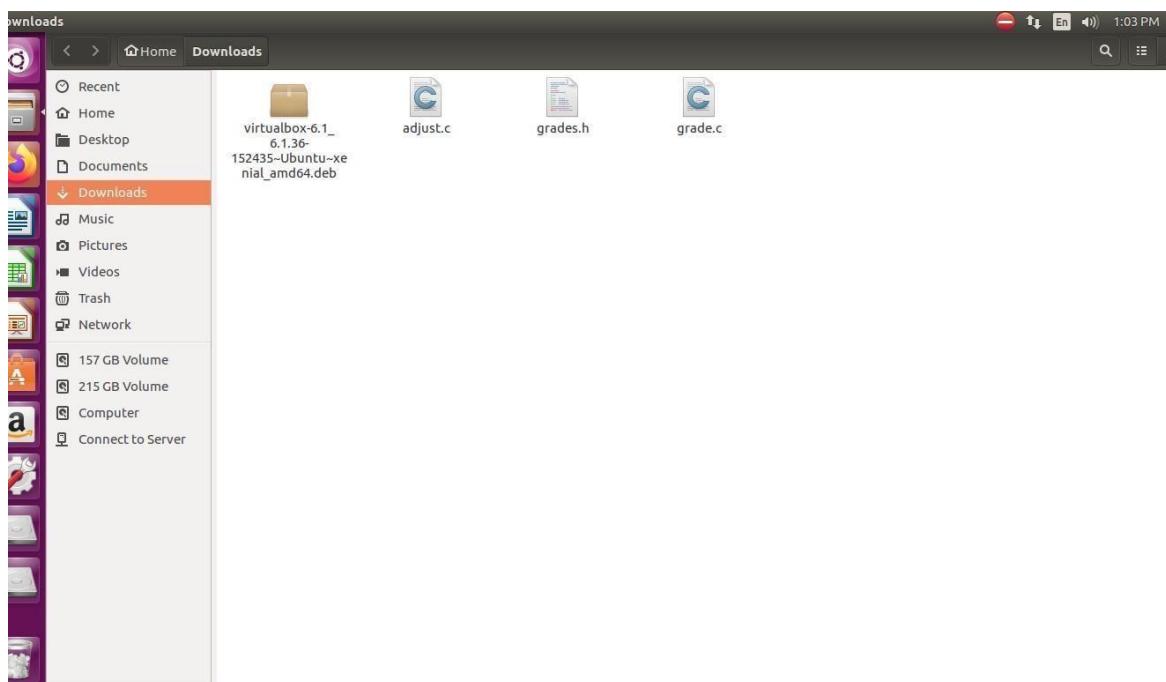
2. **Click Download VirtualBox .**



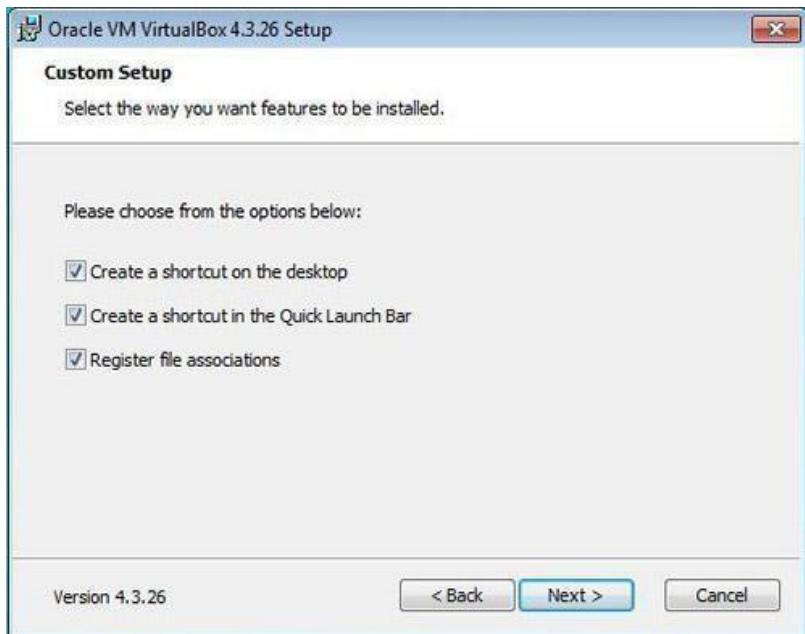
3. **Click Windows hosts.** You'll see this link below the "VirtualBox 6.1.14 platform packages" heading. The VirtualBox EXE file will begin downloading onto your computer and saved in your specified folder.



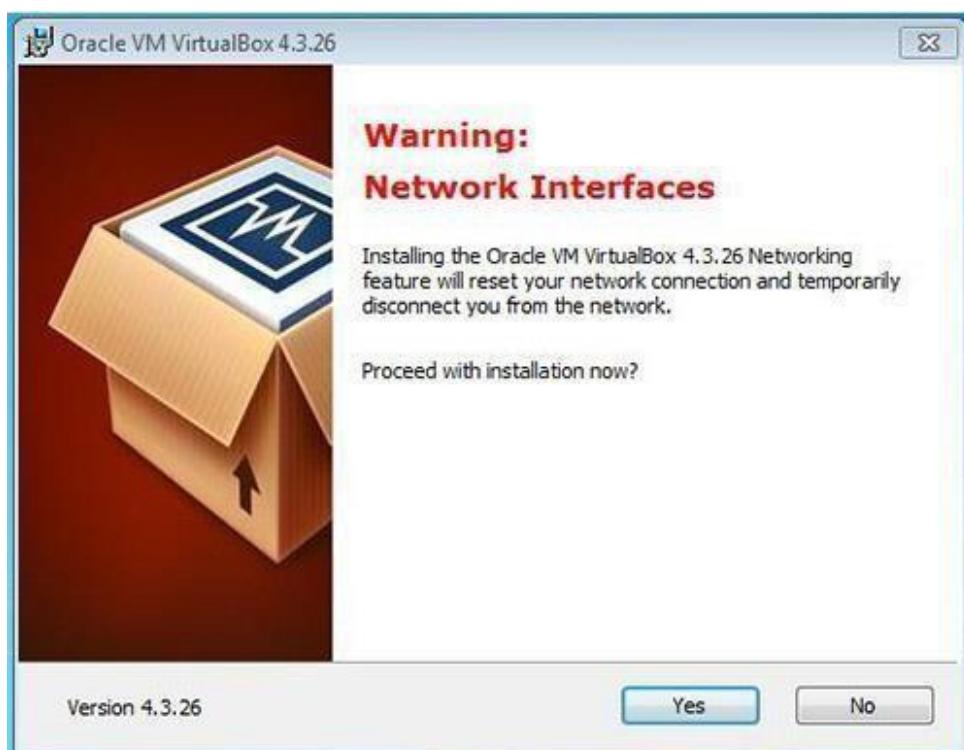
4. **Open the VirtualBox EXE file.** Go to the location to which the EXE file downloaded and double-click the file.



5. Navigate through the installation prompts. Do the following:
 - i) Click Next on the first three pages.



- ii) Click Yes when prompted.



6. **Click Install when prompted.** Doing so will allow VirtualBox to begin installing on your computer.



7. **Click Finish when prompted.**



Creating a Virtual Machine :

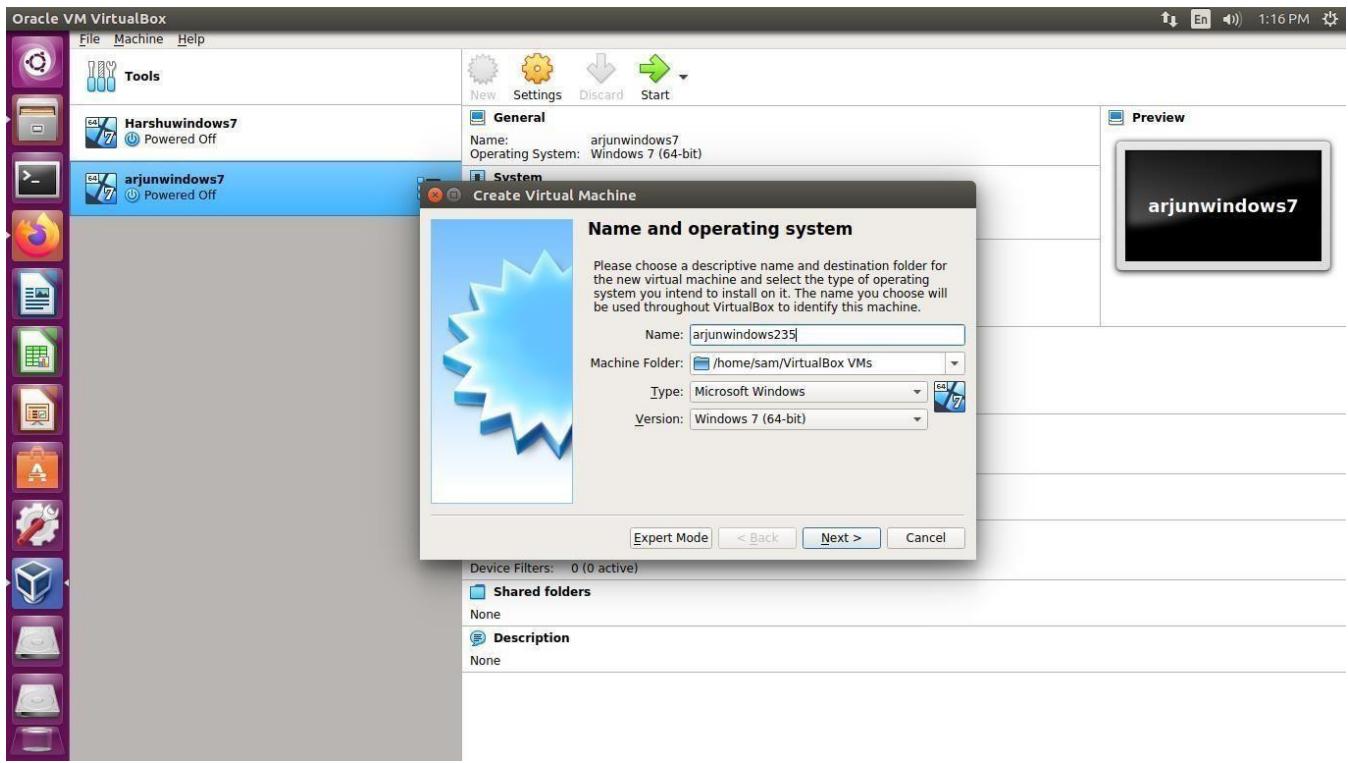
1. To create a new virtual machine, you need to start VirtualBox. On the host where you installed Oracle VDI and VirtualBox, select the **Applications** menu on the desktop, then the **System Tools** menu, and then **Oracle VM VirtualBox**. Alternatively, you can run the **VirtualBox** command in a terminal. The Oracle VM VirtualBox Manager is displayed



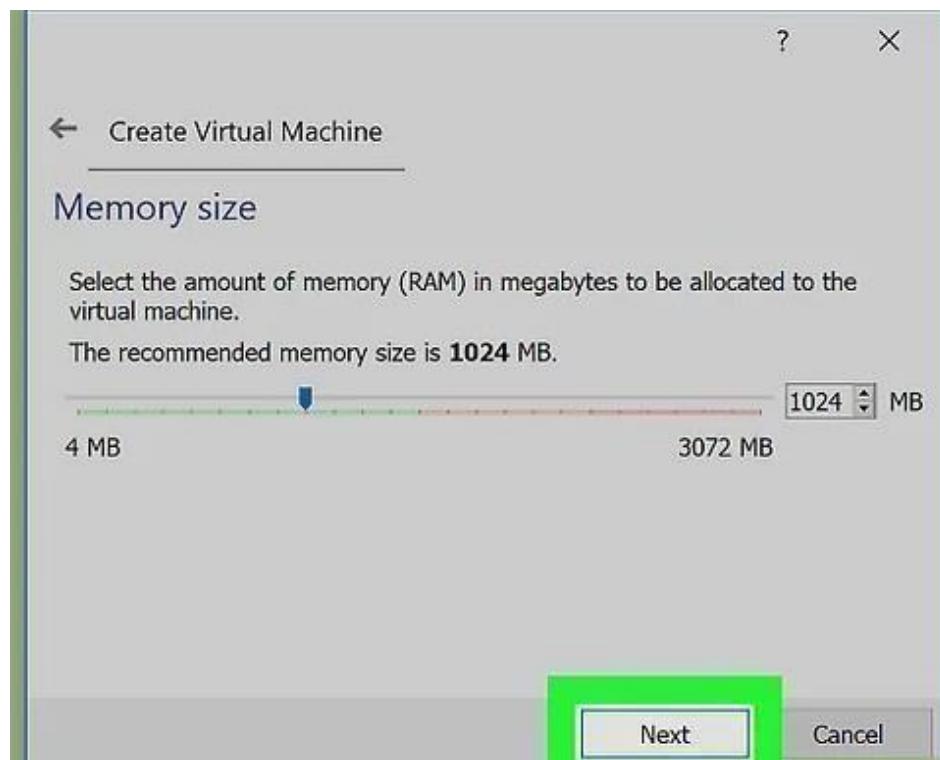
2. Click the **New** button. The New Virtual Machine Wizard is displayed in a new window.



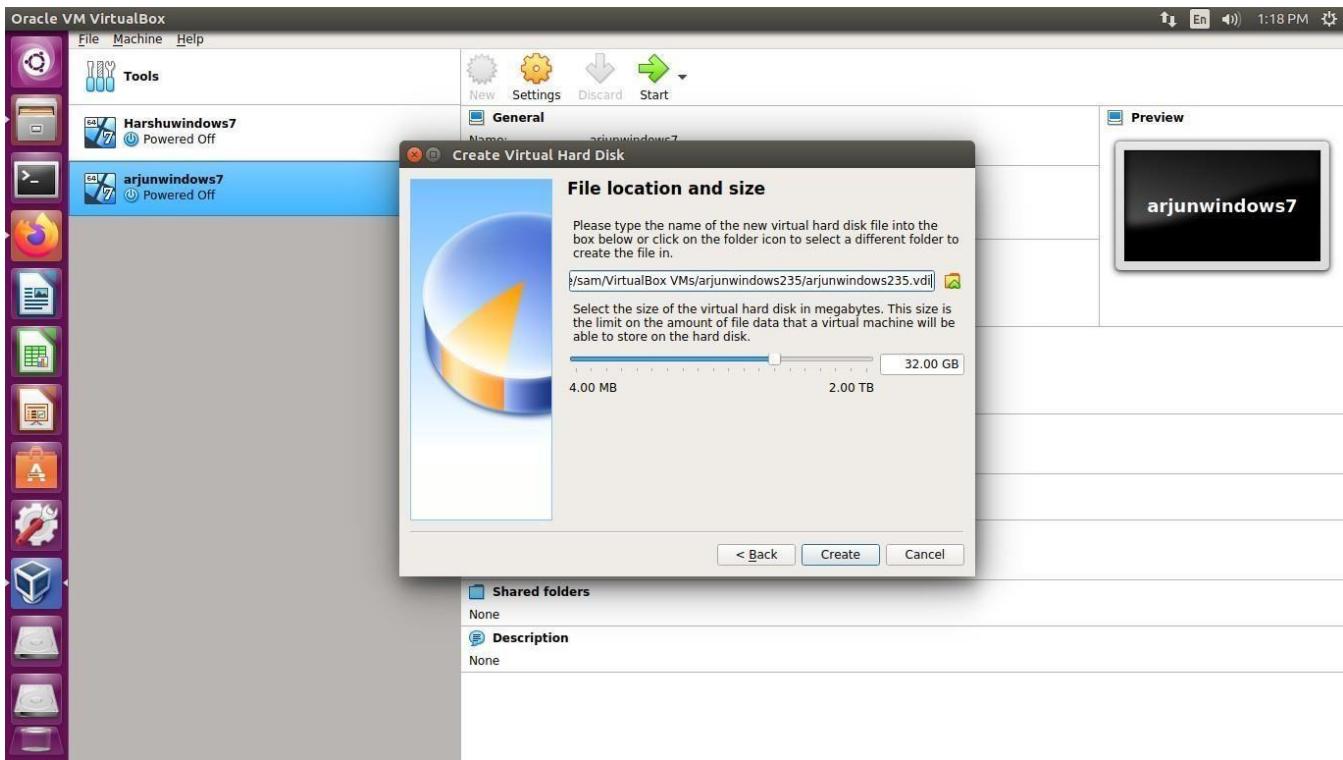
3. Click the **Next** button ,The wizard enables you to configure the basic details of the virtual machine. On the VM Name and OS Type step, enter a descriptive name for the virtual machine in the **Name** field and select the operating system and version that you are going to install from the drop-down lists.



4. On the Memory step, you can simply accept the default.



5. On the Virtual Hard Disk step, ensure **Start-up Disk** is selected . select **Create new hard disk** and click **Next**.

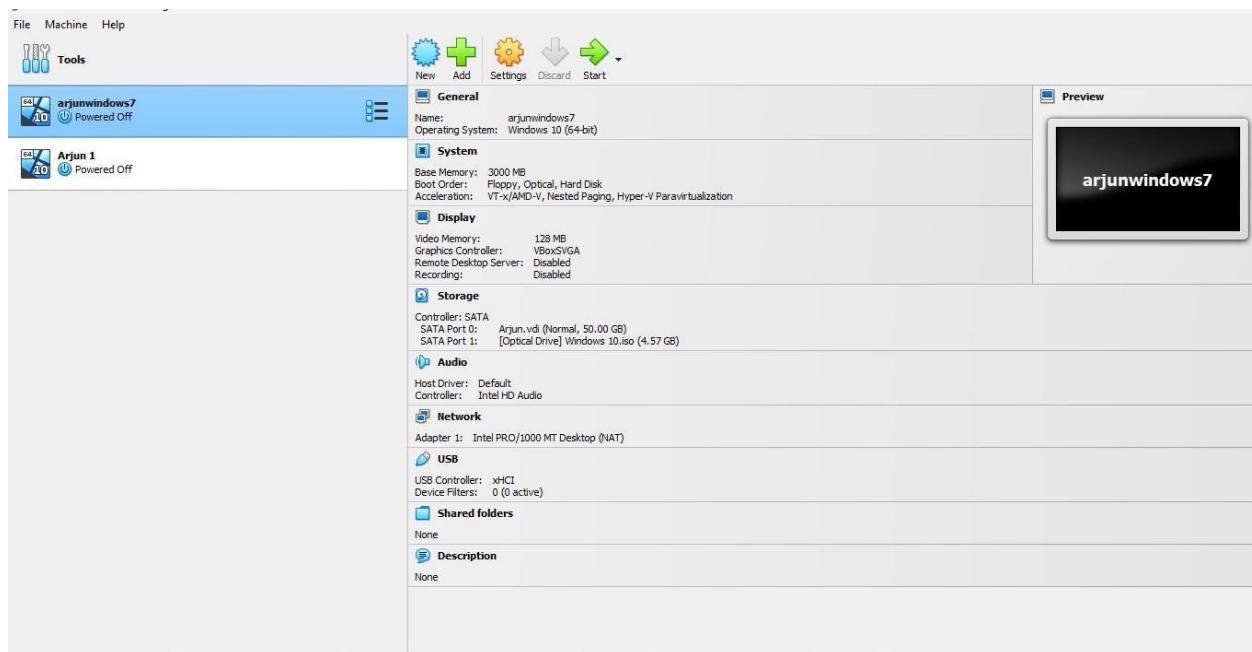


6. Select **VDI (VirtualBox Disk Image)** as the file type, **Dynamically allocated** as the storage details, and accept the defaults for the virtual disk file location and size, and then click **Create** to create the virtual disk.





- When the virtual disk is created, the Virtual Disk Creation Wizard is closed and you are returned to the Summary step of the New Virtual Machine Wizard. Click **Create** to create the virtual machine. The wizard is closed and the newly-created virtual machine is listed in Oracle VM VirtualBox Manager,

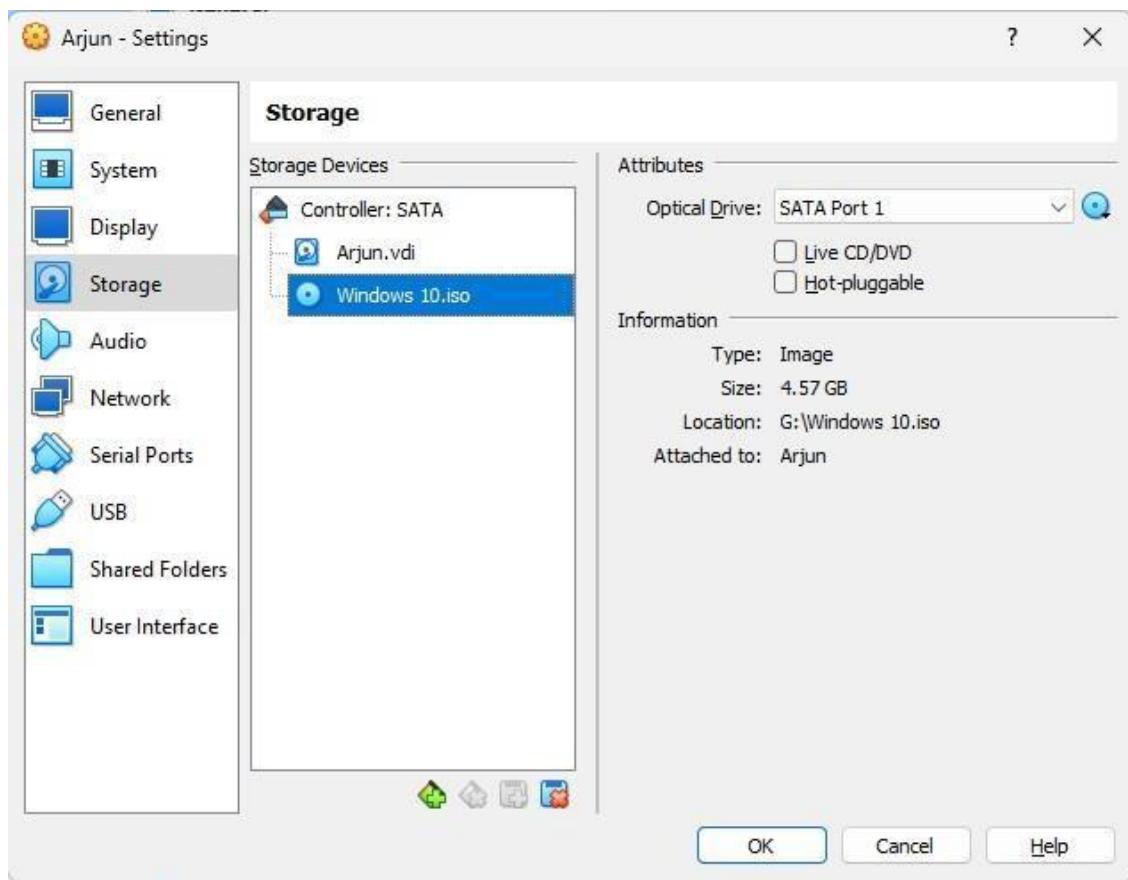


8. Since you want to install an operating system in the virtual machine, you need to make sure the virtual machine can access the installation media. To do this, you edit the virtual machine settings. In Oracle VM VirtualBox Manager, select the virtual machine and then in the toolbar click the **Settings** button. The Settings window is displayed. In the navigation on the left, select **Storage**.

In the Storage Tree section, select **Empty** below the IDE Controller. The CD/DVD Drive attributes are displayed. Click the CD/DVD icon next to the **CD/DVD Drive** drop-down list and select the location of the installation media, as follows:

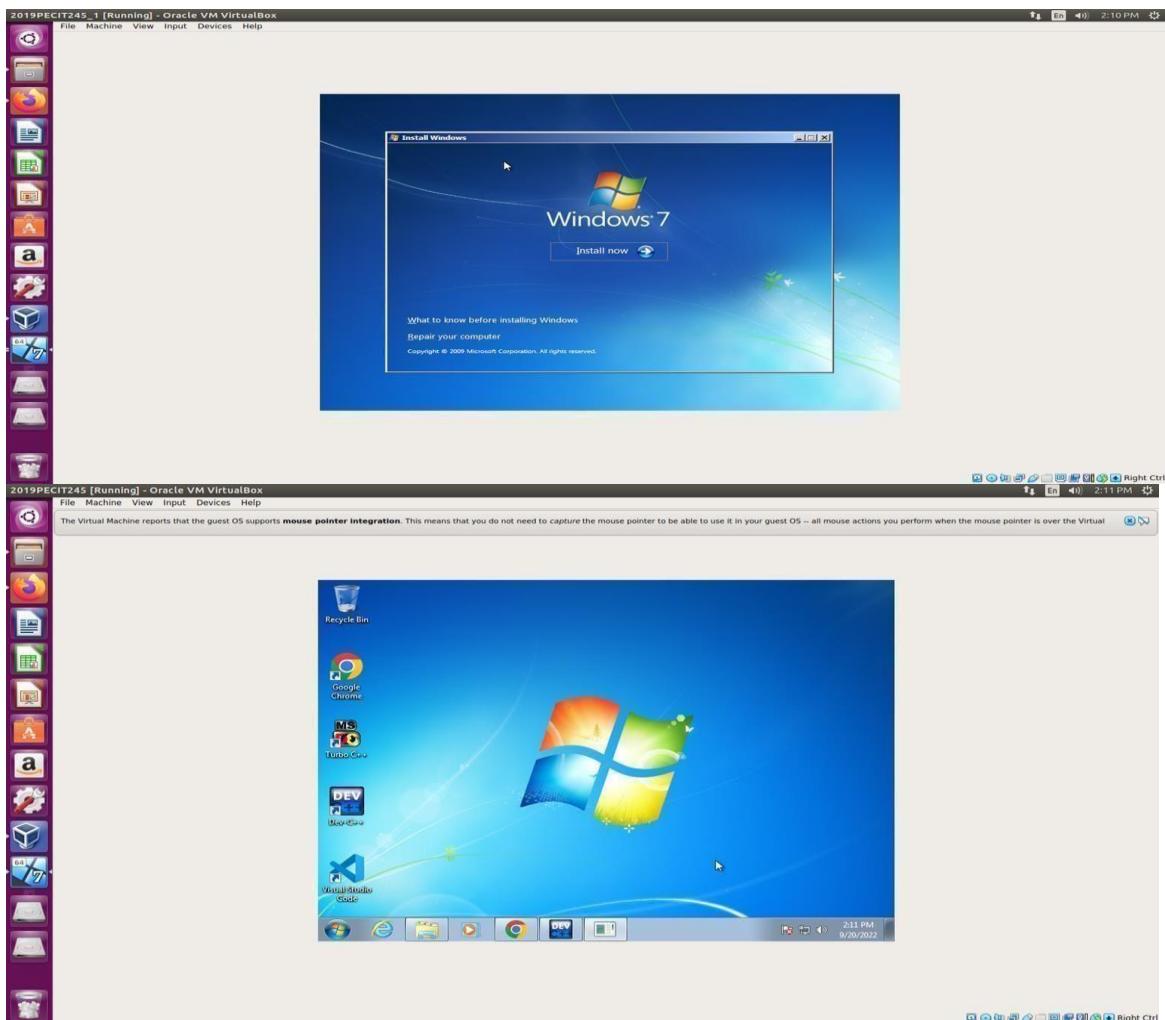
- To connect the virtual CD/DVD drive to the host's physical CD/DVD drive, select **Host Drive <drive-name>**.

To insert an ISO image in the virtual CD/DVD drive, select **Choose a virtual CD/DVD disk file** and browse for the ISO image.



9. Click **OK** to apply the storage settings. You are now ready to start the virtual machine and install the operating system. In Oracle VM VirtualBox Manager, select the virtual machine and click the **Start** button in the toolbar. A new window is displayed, which shows the virtual machine booting up.

10. Start the operating system installation.



Result :

The procedure to Install Virtualbox/VMware Workstation with different flavors of linux or windows OS on top of windows7 or 8 was learned and verified successfully .

**FIND PROCEDURE TO RUN THE VIRTUAL MACHINE OF
DIFFERENT CONFIGURATION. CHECK HOW MANY VIRTUAL
MACHINES CAN BE UTILIZED AT PARTICULAR TIME.**

Aim :

To Find procedure to run the virtual machine of different configuration. Check how many virtual machines can be utilized at particular time using Open Nebula and Virtual Machine Manager.

Procedure :

Creating a guest virtual machine with virt-manager

1. Creating a guest virtual machine with virt-manager

i. Open virt-manager

Start virt-manager. Launch the **Virtual Machine Manager** application from the **Applications** menu and **System Tools** submenu. Alternatively, run the virt-manager command as root.

ii. Optional: Open a remote hypervisor

Select the hypervisor and click the **Connect** button to connect to the remote hypervisor.

iii. Create a new virtual machine

The **virt-manager** window allows you to create a new virtual machine. Click the **Create a new virtual machine** button to open the **New VM** wizard.

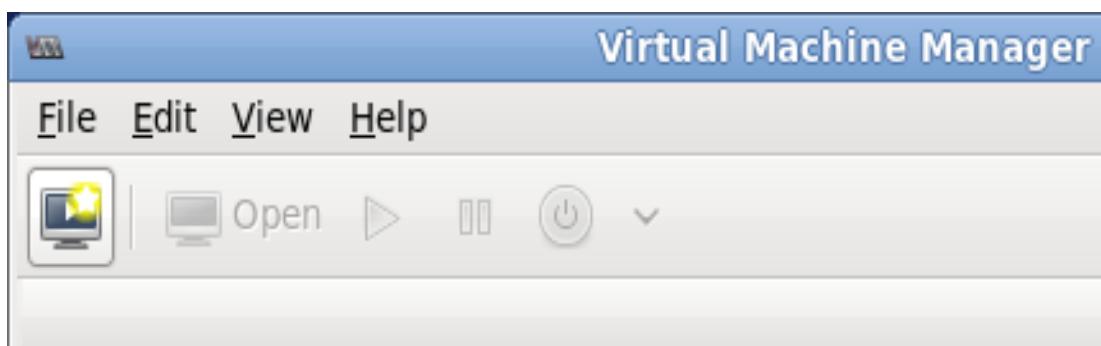


Figure : Virtual Machine Manager window

The **New VM** wizard breaks down the virtual machine creation process into five steps:

1. Naming the guest virtual machine and choosing the installation type
2. Locating and configuring the installation media
3. Configuring memory and CPU options
4. Configuring the virtual machine's storage

5. Configuring networking, architecture, and other hardware settings

Ensure that virt-manager can access the installation media (whether locally or over the network) before you continue.

iv. Specify name and installation type

The guest virtual machine creation process starts with the selection of a name and installation type. Virtual machine names can have underscores (_), periods (.), and hyphens (-).

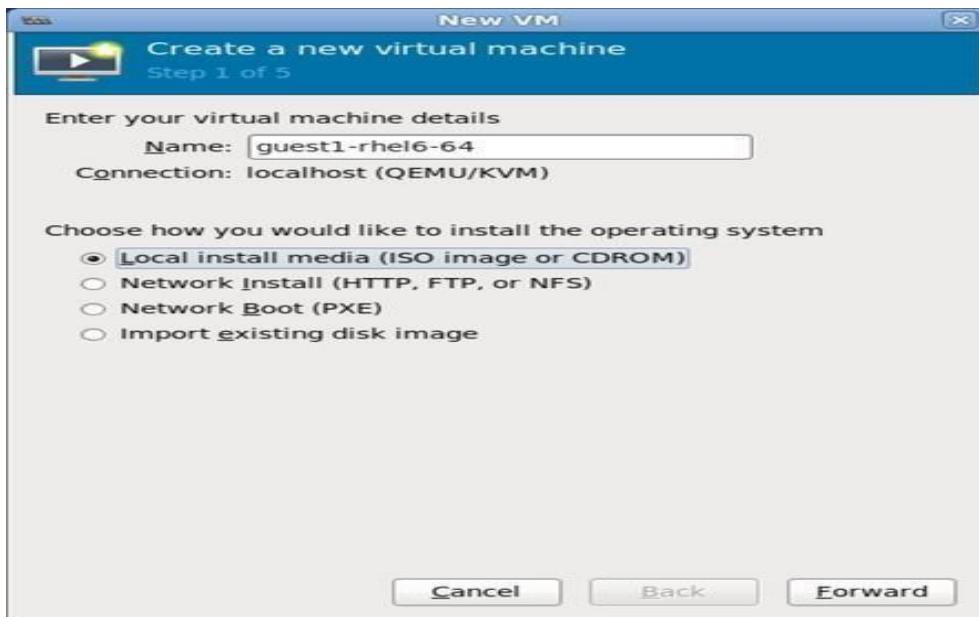


Figure : Name virtual machine and select installation method

Type in a virtual machine name and choose an installation type:

Local install media (ISO image or CDROM)

This method uses a CD-ROM, DVD, or image of an installation disk (for example, .iso).

Network Install (HTTP, FTP, or NFS)

This method involves the use of a mirrored Red Hat Enterprise Linux or Fedora installation tree to install a guest. The installation tree must be accessible through either HTTP, FTP, or NFS.

Import existing disk image

This method allows you to create a new guest virtual machine and import a disk image (containing a pre-installed, bootable operating system) to it.

Click **Forward** to continue.

2. Configure installation

Next, configure the **OS type** and **Version** of the installation. Ensure that you select the appropriate OS type for your virtual machine.

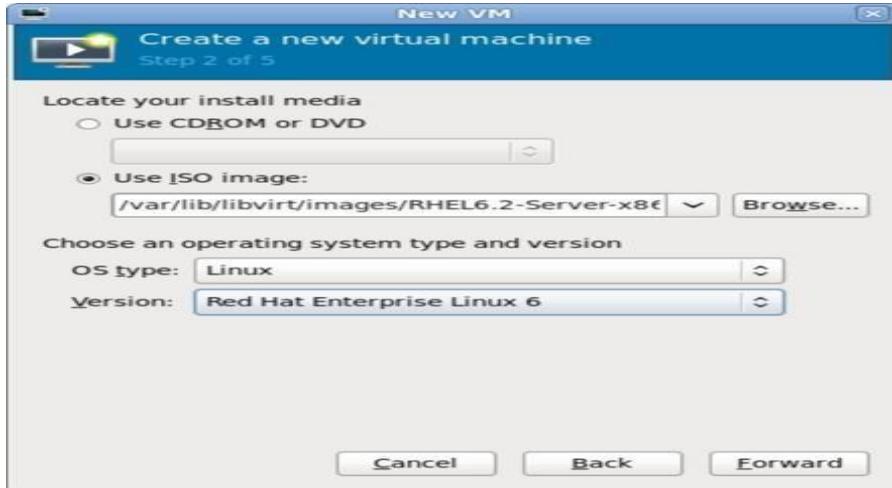


Figure : Local ISO image installation

i. Configure CPU and memory

The next step involves configuring the number of CPUs and amount of memory to allocate to the virtual machine. The wizard shows the number of CPUs and amount of memory you can allocate; configure these settings and click **Forward**.

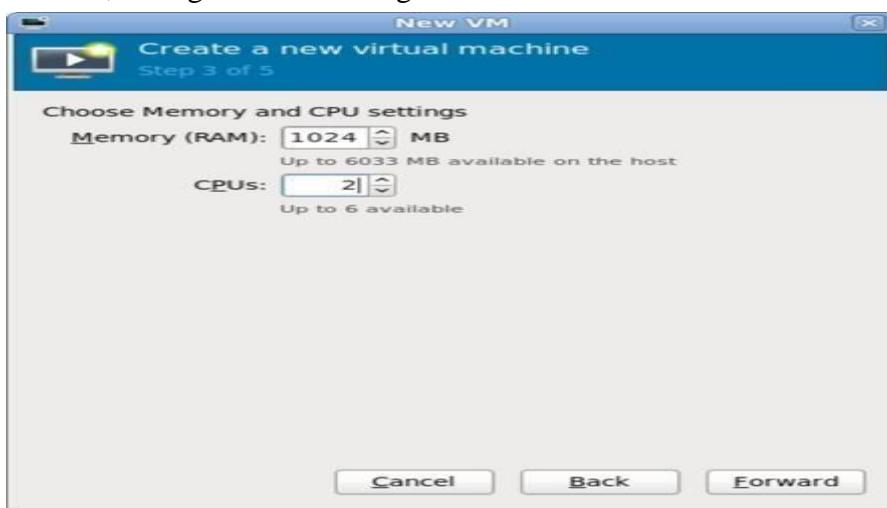


Figure : Configuring CPU and memory

Configure storage

Assign storage to the guest virtual machine.

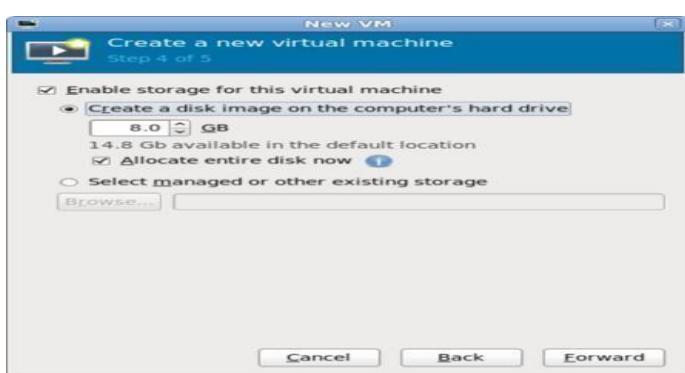


Figure : Configuring virtual storage

If you chose to import an existing disk image during the first step, virt-manager will skip this step. Assign sufficient space for your virtual machine and any applications it requires, then click **Forward** to continue.

7. Final configuration

Verify the settings of the virtual machine and click **Finish** when you are satisfied; doing so will create the virtual machine with default networking settings, virtualization type, and architecture.



Figure : Verifying the configuration

If you prefer to further configure the virtual machine's hardware first, check the **Customize configuration before install** box first before clicking **Finish**. Doing so will open another wizard that will allow you to add, remove, and configure the virtual machine's hardware settings.

After configuring the virtual machine's hardware, click **Apply**. virt-manager will then create the virtual machine with your specified hardware settings.

Installing KVM

KVM only works if your CPU has hardware virtualization support – either Intel VT-x or AMD-V. To determine whether your CPU includes these features, run the following command:

1 . egrep -c '(svm|vmx)' /proc/cpuinfo

0 indicates that your CPU doesn't support hardware virtualization , **1** enable hardware virtualization support in your computer's BIOS.

Virt-Manager is a graphical application for managing your virtual machines

2. sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager

3. sudo adduser panimalar libvirtd

In root user

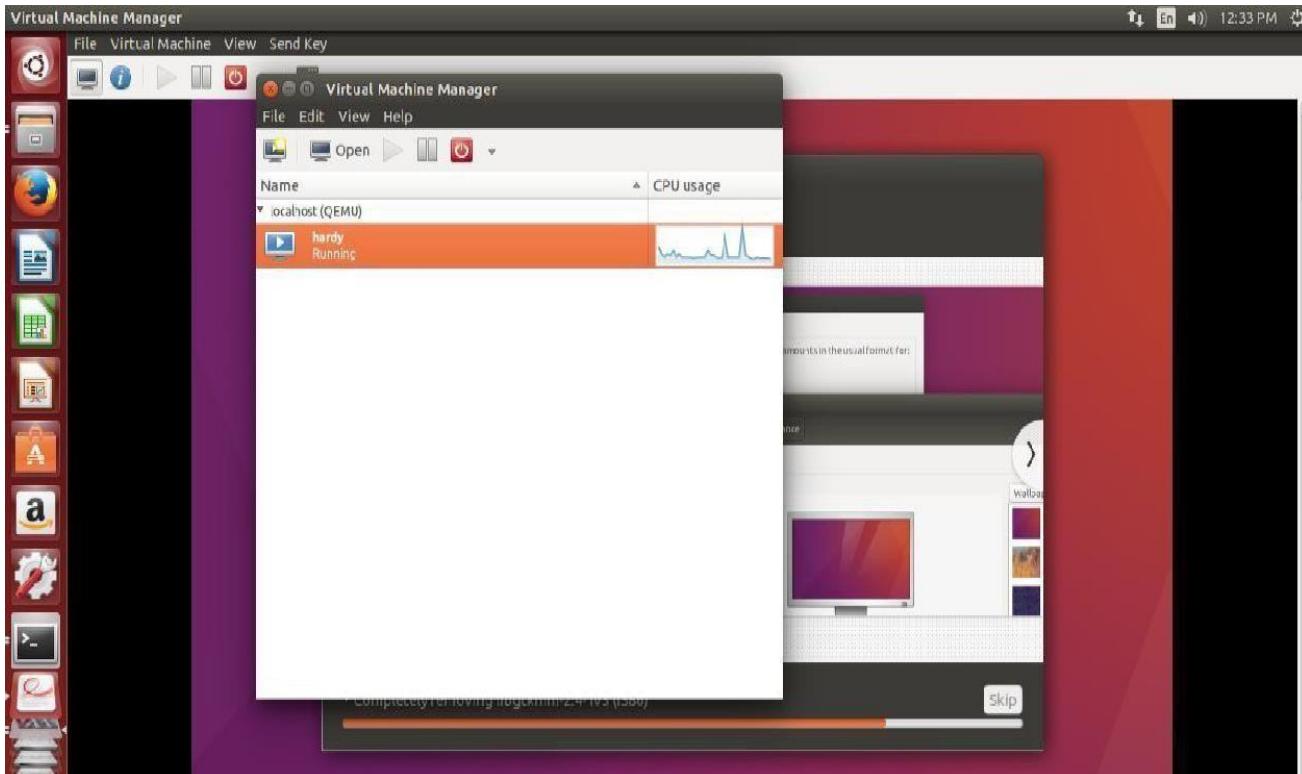
4. irsh -c qemu:///system list

| Id | Name | State |
|----|------|-------|
|----|------|-------|

This indicates that everything is working correctly.

5. sudo virt-manager

Sample Output:



Result :

The procedure to run the virtual machine of different configuration. Check how many virtual machines can be utilized at particular time using Open Nebula and Virtual Machine Manager was successfully created and verified .

**INSTALL A C COMPILER IN THE VIRTUAL MACHINE
CREATED USING VIRTUAL BOX AND EXECUTE
ARITHMETIC OPERATION PROGRAM**

Aim :

To install a C compiler and execute a sample program in the created virtual machine.

Procedure :

1. Start the process.
2. To login into Guest OS in KVM(Kernel Virtual Machine).
3. The most well-known **C** and **C++** compilers are **gcc** and **g++**
4. Installing **C**, **C++** Compiler and Development Tools (build-essential).
5. To login into root user :

```
$ sudo -s  
# apt-get update && apt-get install build-essential  
( OR )  
$ sudo get update && apt-get install build-essential  
( OR )
```

\$ sudo apt-get install gcc

6. Speeding Up **C** and **C++** Compilations

aptitude install ccache

7. Open your favorite text editor and enter the code for adds two numbers, then save as **sum.c**
8. To compile the code into an executable named “**sum**” in the current working directory.

gcc sum.c -o sum

9. If you want to take advantage of **ccache**, just propend the above command with **ccache**, as follows:

ccache gcc sum.c -o sum

10. To run the binary:

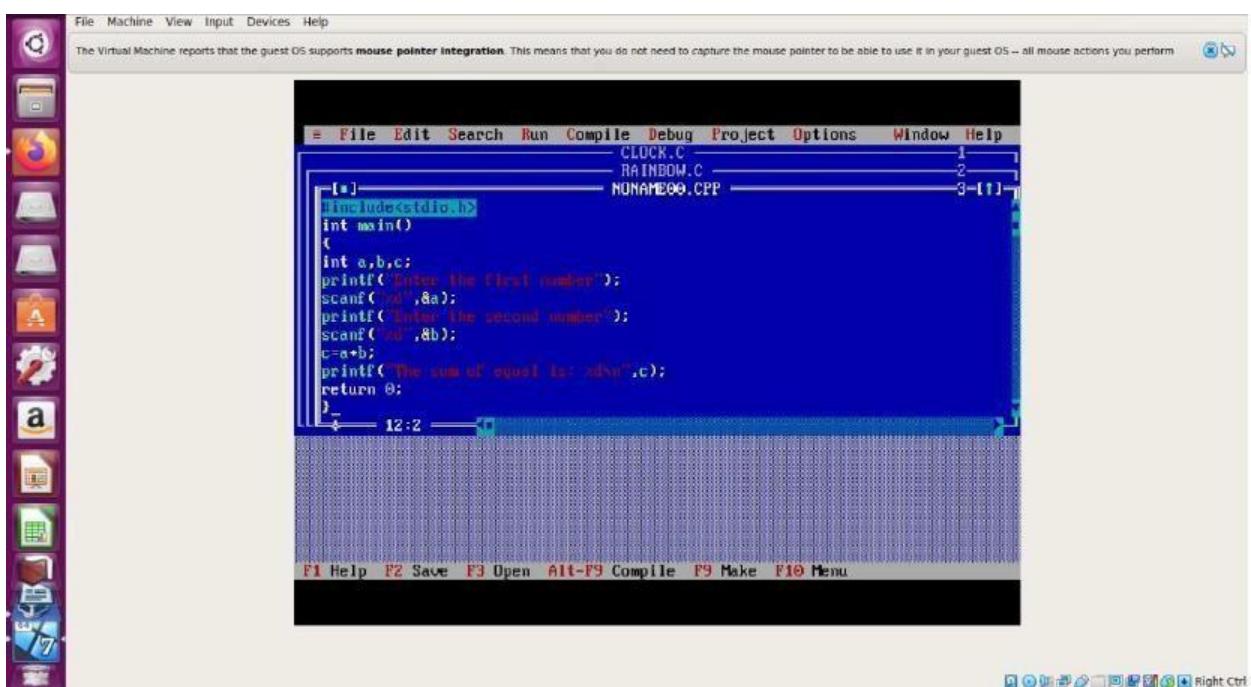
./sum

11. Stop the process .

Program :

```
#include<stdio.h>
int main()
{
    int a, b, c;
    printf("enter first number");
    scanf("%d",&a);
    printf("enter second number");
    scanf("%d",&b);
    c = a + b;
    printf("The sum of equals %d\n",c);
    return 0;
}
```

Sample Input and Output :



The screenshot shows a virtual machine interface with a toolbar on the left containing icons for various applications like file manager, browser, terminal, and settings. The main window has a menu bar with File, Machine, View, Input, Devices, Help. A status bar at the top right says "The Virtual Machine reports that the guest OS supports mouse pointer integration. This means that you do not need to capture the mouse pointer to be able to use it in your guest OS – all mouse actions you perform..." Below the menu is a file list with CLOCK.C, RAINBOW.C, and NONAME00.CPP. The central area displays a C program source code:

```
#include<stdio.h>
int main()
{
    int a,b,c;
    printf("Enter the first number:");
    scanf("%d",&a);
    printf("Enter the second number:");
    scanf("%d",&b);
    c=a+b;
    printf("The sum is %d",c);
    return 0;
}
```

A message box titled "Compiling" shows the compilation results:

```
Main file: NONAME00.CPP
Compiling: EDITOR + NONAME00.CPP
Total      File
Lines compiled: 319   319
Warnings: 0       0
Errors: 0       0
```

Available memory: 1969K

Success : Press any key

The bottom of the screen has a keyboard layout and function keys F1 through F10.

This screenshot shows the same virtual machine environment as the previous one, but the central window now displays the output of the executed C program. The user has entered two numbers, 3 and 4, and the program outputs their sum, 7.

```
Enter the first number:3
Enter the second number:4
The sum of equal is: 7
Enter the first number:4
Enter the second number:5
The sum of equal is: 9
Enter the first number:...
```

The bottom of the screen has a keyboard layout and function keys F1 through F10.

Result :

Thus the installation of a C compiler and execute a sample program in the created using virtual machine was successfully executed and verified

Exp. No. : **4 b**
Date: **20-09-2022**

INSTALL A C COMPILER IN THE VIRTUAL MACHINE CREATED USING VIRTUAL BOX AND FIND SUM OF DIGITS OF GIVEN NUMBER

Aim :

To install a C compiler and execute a sample program in the created virtual machine.

Procedure :

1. Start the process.
2. To login into Guest OS in KVM(Kernel Virtual Machine).
3. The most well-known **C** and **C++** compilers are **gcc** and **g++**
4. Installing C, C++ Compiler and Development Tools (build-essential).
5. To login into root user :

```
$ sudo -s  
# apt-get update && apt-get install build-essential
```

(OR)

```
$ sudo get update && apt-get install build-essential  
( OR )
```

```
$ sudo apt-get install gcc
```

6. Speeding Up C and C++ Compilations

```
# aptitude install ccache
```

7. Open your favorite text editor and enter the code for adds two numbers, then save as sum.c
8. To compile the code into an executable named “**sum**” in the current working directory.

```
# gcc sum.c -o sum
```

9. If you want to take advantage of **ccache**, just propend the above command with ccache, as follows:

```
# ccache gcc sum.c -o sum
```

10. To run the binary:

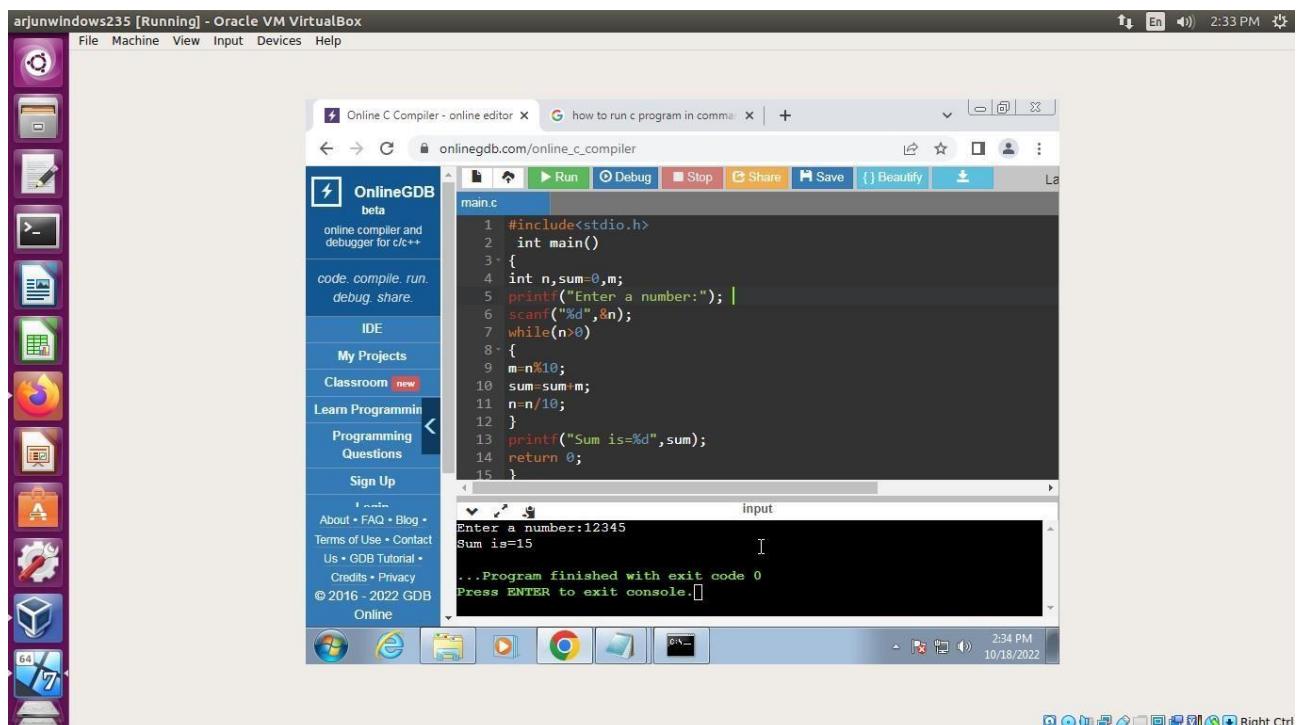
```
# ./sum
```

11. Stop the process .

Program :

```
#include<stdio.h>
int main()
{
    int n, sum=0, m;
    printf("Enter a number: ");
    scanf("%d", &n);
    while(n>0)
    {
        m = n% 10;
        sum = sum + m;
        n = n/10;
    }
    printf("Sum is = %d",sum);
    return 0;
}
```

Sample Input and Output:



Result :

Thus the installation of a C compiler and execute a sample program in the created using virtual machine was successfully executed and verified .

Aim :

To Install Google App Engine. Create *hello world* app and other simple web applications using python/java.

Procedure :

The App Engine SDK allows you to run Google App Engine Applications on your local computer. It simulates the run-time environment of the Google App Engine infrastructure.

Pre-Requisites: Python 2.5.4

If you don't already have Python 2.5.4 installed in your computer, download and Install Python 2.5.4 from:

<http://www.python.org/download/releases/2.5.4/>

Download and Install

You can download the Google App Engine SDK by going to:

<http://code.google.com/appengine/downloads.html>

and download the appropriate install package.

Download the Google App Engine SDK

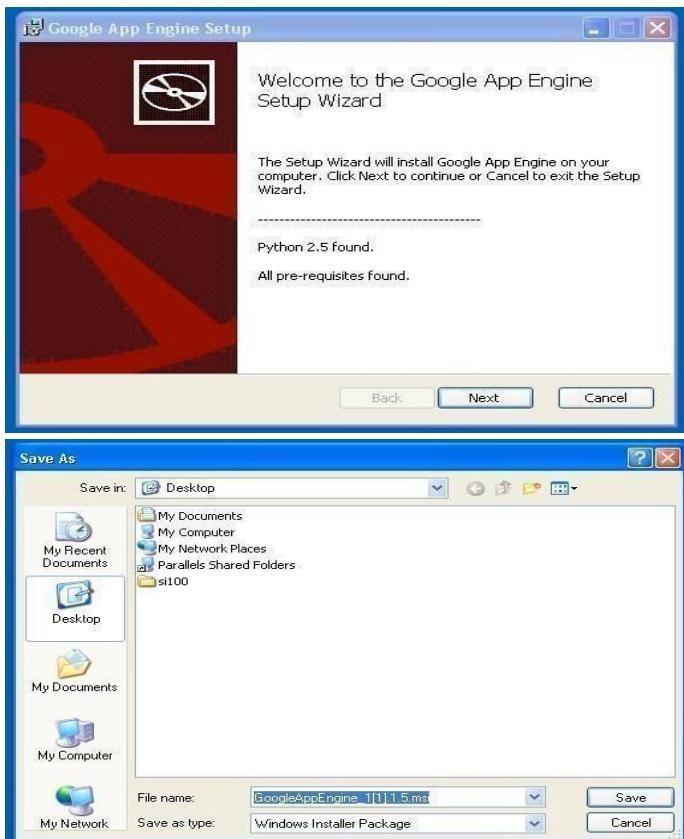
Before downloading, please read the [Terms](#) that govern your use of the App Engine SDK.

Please note: The App Engine SDK is under **active development**, please keep this in mind as you explore its capabilities. See the [SDK Release Notes](#) for the information on the most recent changes to the App Engine SDK. If you discover any issues, please feel free to notify us via our [Issue Tracker](#).

| Platform | Version | Package | Size | SHA1 Checksum |
|-----------------------|------------------|---|--------|--|
| Windows | 1.1.5 - 10/03/08 | GoogleAppEngine_1.1.5.msi | 2.5 MB | e974312b4aefc0b3873ff0d93eb4c525d5e88c30 |
| Mac OS X | 1.1.5 - 10/03/08 | GoogleAppEngineLauncher-1.1.5.dmg | 3.6 MB | f62208ac01c1b3e39796e58100d5f1b2f052d3e7 |
| Linux/Other Platforms | 1.1.5 - 10/03/08 | google_appengine_1.1.5.zip | 2.6 MB | cbb9ce817bdabf1c4f181d9544864e55ee253de1 |

Download the Windows installer – the simplest thing is to download it to your Desktop or another folder that you remember.

Double Click on the **GoogleApplicationEngine** installer.



Click through the installation wizard, and it should install the App Engine. If you do not have Python 2.5, it will install Python 2.5 as well.

Once the install is complete you can discard the downloaded installer



Making your First Application

- **Create a project**

Projects bundle code, VMs, and other resources together for easier development and monitoring.

- **Build and run your "Hello World!" app**

You will learn how to run your app using Cloud Shell, right in your browser. At the end, you'll deploy your app to the web using the gcloud command.

Step 1 : Project setup

GCP organizes resources into projects, which collect all of the related resources for a single application in one place.

Step 2 : Using Cloud Shell

Cloud Shell is a built-in command-line tool for the console. We're going to use Cloud Shell to deploy our app.

Open Cloud Shell

Open Cloud Shell by clicking the **Activate Cloud Shell** button in the navigation bar in the upper-right corner of the console.

Clone the sample code

Use Cloud Shell to clone and navigate to the "Hello World" code. The sample code is cloned from your project repository to the Cloud Shell.

In Cloud Shell, enter the following:

```
$ git clone \ https://github.com/GoogleCloudPlatform/python-docs-samples  
$ cd \ python-docs-samples/appengine/standard_python3/hello_world
```

Step 3 : Configuring your deployment

You are now in the main directory for your code.

Exploring the application :

```
$ cat main.py
```

```
# [START gae_python38_app]  
from flask import Flask  
  
# If `entrypoint` is not defined in app.yaml, App Engine will look for an app called `app` in  
`main.py`.  
app = Flask(__name__)  
@app.route('/')  
def hello():  
    """Return a friendly HTTP greeting."""  
    return 'Hello World!'  
if __name__ == '__main__':  
    # This is used when running locally only. When deploying to Google App
```

```
# Engine, a webserver process such as Gunicorn will serve the app. This  
# can be configured by adding an `entrypoint` to app.yaml.  
app.run(host='127.0.0.1', port=8080, debug=True)  
# [END gae_python38_app]
```

The application is a simple Python application that uses the Flask web framework. This Python app responds to a request with an HTTP header and the message Hello World!.

Exploring your configuration :

\$ cat app.yaml

This file contains the minimal amount of configuration required for a Python 3 application.

The runtime field specifies the python38 run-time environment.

```
runtime: python38
```

Example :

```
runtime: python38      # or python37 for Python 3.7  
instance_class: F2  
env_variables:  
  BUCKET_NAME: "example-gcs-bucket"  
handlers:  
  # Matches requests to /images/... to files in static/images/...  
  - url: /images  
    static_dir: static/images  
  
  - url: /.*  
    secure: always  
    redirect_http_response_code: 301  
    script: auto
```

Step 4 : Testing your app

Cloud Shell lets you test your app before deploying to make sure it's running as intended, just like debugging on your local machine.

To test your app, first create an isolated virtual environment. This ensures that your app does not interfere with other Python applications that may be available on your system.

\$ virtualenv --python python3 ~/envs/hello_world

Activate your newly created virtual environment:

\$ source ~/envs/hello_world/bin/activate

Use pip to install project dependencies. This "Hello World" app depends on the Flask microframework:

\$ pip install -r requirements.txt

Finally, run your app in Cloud Shell using the Flask development server:

```
$ python main.py
```

Preview your app with "Web preview"

Your app is now running on Cloud Shell. You can access the app by clicking the **Web preview** button at the top of the Cloud Shell pane and choosing **Preview on port 8080**.

Sample output :



Terminating the preview instance

Terminate the instance of the application by pressing Ctrl+C in the Cloud Shell.

Step 5 : Deploying to App Engine

Create an application :

In order to deploy your app, you need to create an app in a region:

```
$ gcloud app create
```

Note: If you already created an app, you can skip this step.

Deploying with Cloud Shell:

You can use Cloud Shell to deploy your app. To deploy your app enter the following:

```
$ gcloud app deploy app.yaml --project adept-rock-295007
```

Visit your app

Congratulations! Your app has been deployed. The default URL of your app is a subdomain on appspot.com that starts with your project's ID: adept-rock-295007.appspot.com

```

target project: [adept-rock-295007]
target service: [default]
target version: [20201108t092724]
target url: [https://adept-rock-295007.uc.r.appspot.com]

Do you want to continue (Y/n)? y

Beginning deployment of service [default]...
[ Uploading 1638 files to Google Cloud Storage ]

File upload done.
Updating service [default]...failed.
ERROR: (gcloud.app.deploy) Error Response: [7] Access Not Configured. Cloud Build has not been used in project adept-rock-295007 before or it is disabled. Enable it by visiting https://console.developers.google.com/apis/api/cloudbuild.googleapis.com/overview?project=adept-rock-295007 then retry. If you enabled this API recently, wait a few minutes for the action to propagate to our systems and retry.

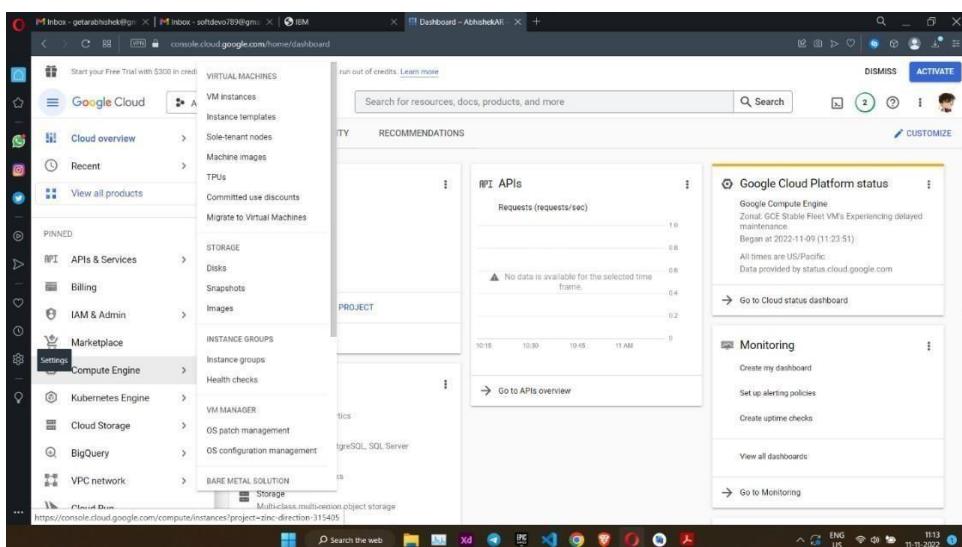
```

Step 6 : View your app's status

You can check in on your app by monitoring its status on the App Engine dashboard.

Open the **Navigation menu** in the upper-left corner of the console.

Then, select the **App Engine** section.



Step 7 : Disable your application

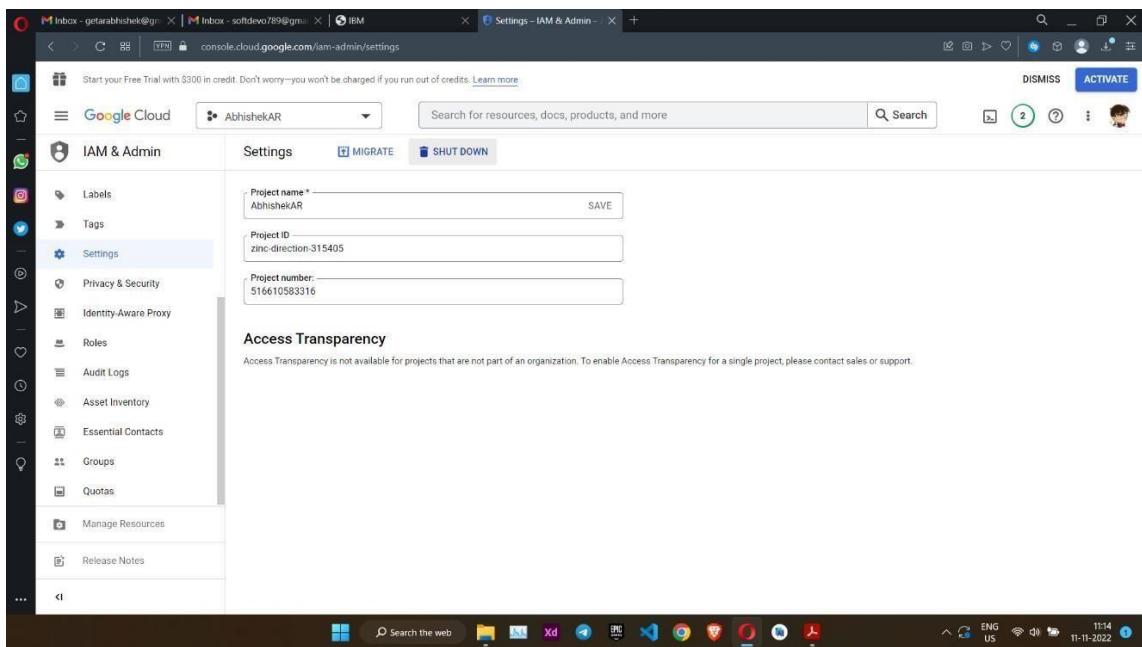
1. Go to the **Settings** page.
2. Click **Disable Application**.

This is sufficient to stop billing from this app. More details on the relationship between apps and projects and how to manage each can be found [here](#).

Delete your project

If you would like to completely delete the app, you must delete the project in the **Manage resources** page. This is not reversible, and any other resources you have in your project will be destroyed:

1. Go to **IAM & admin**
2. Click **Settings**.
3. Click **Shut down**.



Result :

The procedure for installations of Google App Engine. Create hello world app and other simple web applications using python/java was successfully learned and verified .

Exp. No. : 6
Date:10-10-2022

USE GAE LAUNCHER TO LAUNCH THE WEB APPLICATIONS.

Aim :

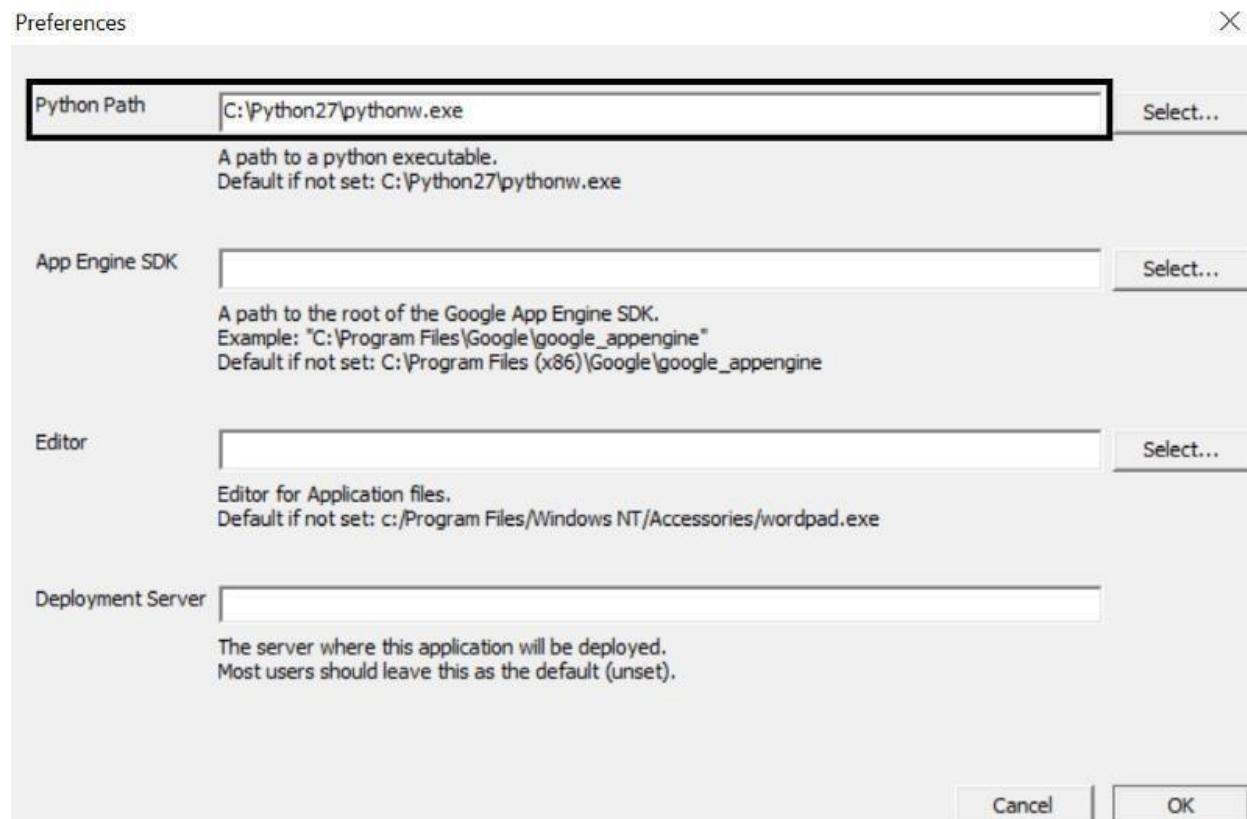
To Use GAE launcher to launch the any web application using python.

Procedure :

Step 1. Download the basic housekeeping stuff

1. Download [Python 2.7](#)
2. Download [Google Cloud SDK](#)
3. Set the Python path in the Google App Engine launcher

After downloading the SDK, launch the App Engine launcher, go to Edit -> Preferences and make sure you set the path for where you installed Python in step 1 above.



Set the Python path in Google App Engine launcher

That's all you need. Your local machine should now be ready to build webapps.

Step 2. App Engine sign-up

This is often the most confusing part of the entire setup. Things you should know when you sign-up:

1. Currently, App Engine offers a free trial for one year.
2. The trial includes \$300 of credit that can be used during the one year trial period.
3. You will need to add a credit card to sign-up (for verification purposes).
4. You will not be charged during the sign-up process.
5. You will not be charged during the trial period as long as you do not cross the credit limit offered.

Here are the steps you need to follow to sign-up:

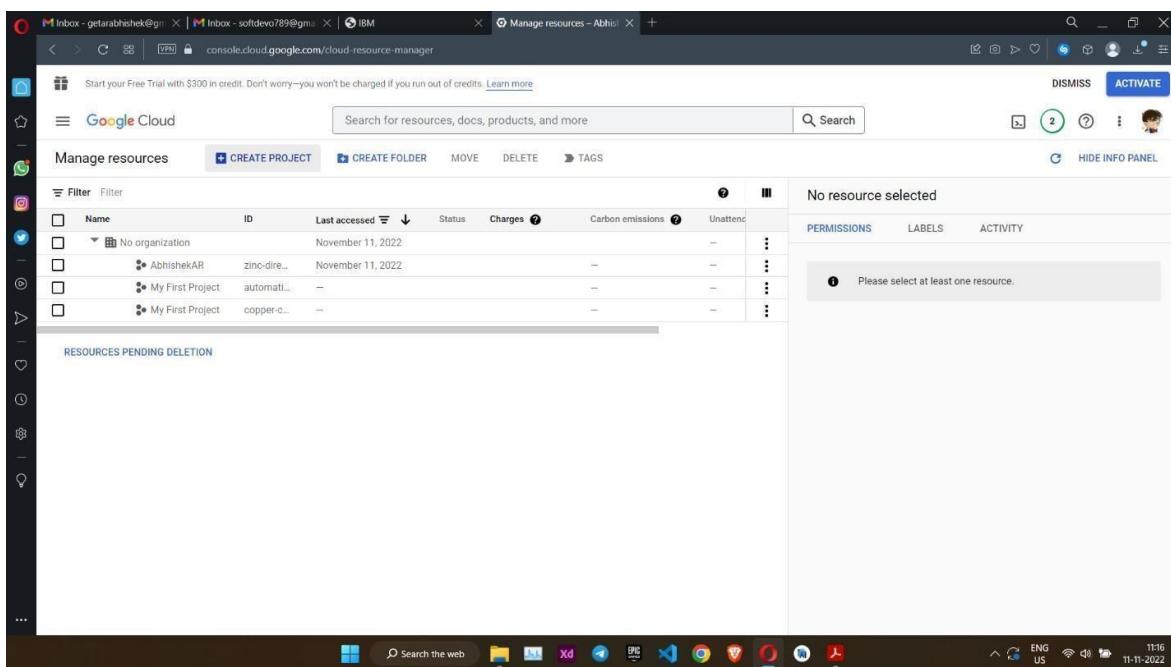
1. Go to the [Google Cloud](#) landing page
2. Follow the sign-up process and go to your App Engine dashboard

Most of the hard work is complete after a successful sign-up.

Step 3. Create a new project

The next step is to create a new Python project that you can work on. Follow the screenshots below to create a new project.

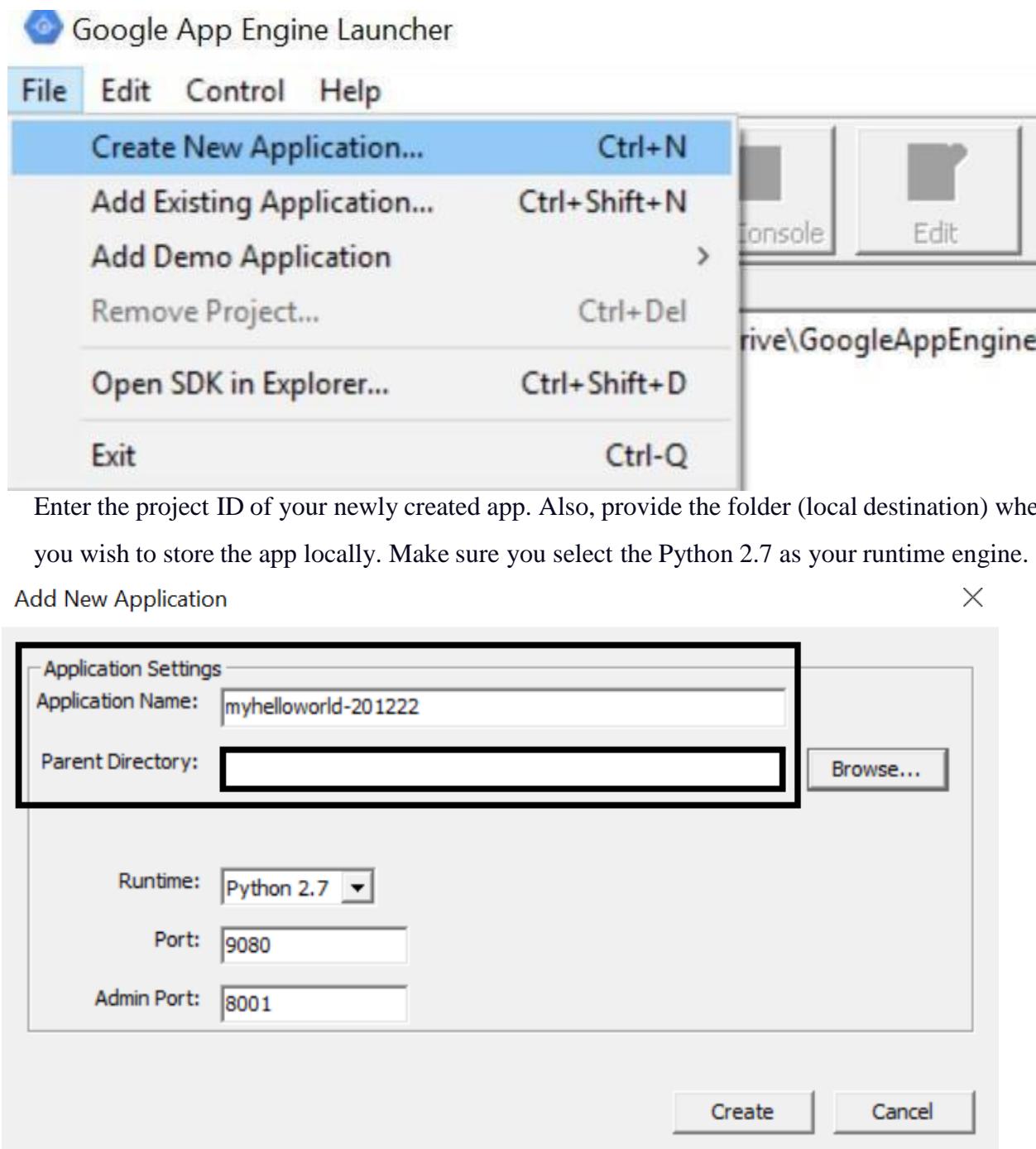
Launch the new project wizard.



Step 4. Fork the app to develop it locally

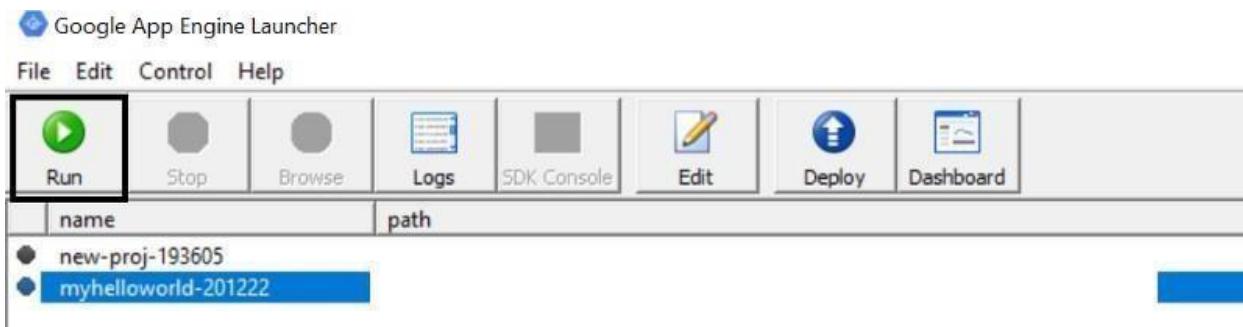
The next step in the process is to fork the app on your local machine. This will allow you to make changes to the app locally and deploy it whenever you wish to.

Go to Google App Engine launcher and create a new application.



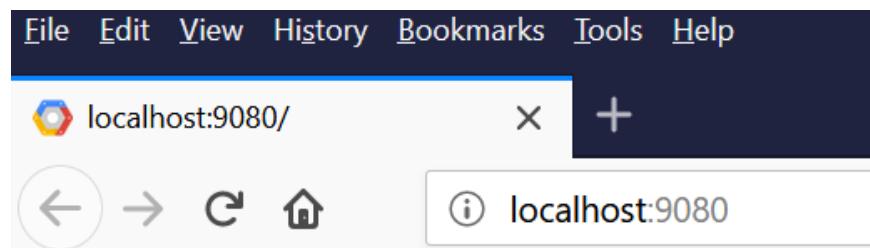
Step 5. Run the app locally

Before you go ahead and make some changes to the app, it is important to check whether or not you have executed all the above steps correctly. This can be done by simply running the app



locally. Select the app and hit the run button on the window.

Wait for a few seconds until you can hit the **Browse** button. Once the **Browse** button becomes clickable, click it. This should take you to the browser, and you should see the hello world text appear in your browser window. Alternatively, you can manually go to the browser and use the port specified to access the app.



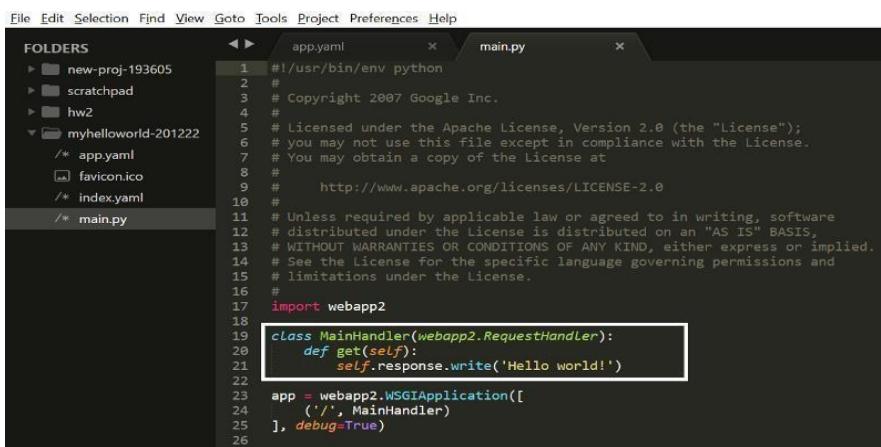
Step 6. Understand the app structure

It is finally time to look at the lines of code which are running this webapp. Open your app folder in the text editor of your choice.

Here is a description of the various files.

app.yaml

This file is a basic markup file that stores information (some metadata) about the app. It is important to note the following crucial parts of the file.



```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS app.yaml main.py
new-proj-193605
scratchpad
hw2
myhelloworld-201222
/* app.yaml
 favicon.ico
/* index.yaml
/* main.py

1 #!/usr/bin/env python
2 #
3 # Copyright 2007 Google Inc.
4 #
5 # Licensed under the Apache License, Version 2.0 (the "License");
6 # you may not use this file except in compliance with the License.
7 # You may obtain a copy of the License at
8 #
9 #     http://www.apache.org/licenses/LICENSE-2.0
10 #
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 # See the License for the specific language governing permissions and
15 # limitations under the License.
16 #
17 import webapp2
18
19 class MainHandler(webapp2.RequestHandler):
20     def get(self):
21         self.response.write('Hello world!')
22
23 app = webapp2.WSGIApplication([
24     ('/', MainHandler)
25 ], debug=True)
```

1. application

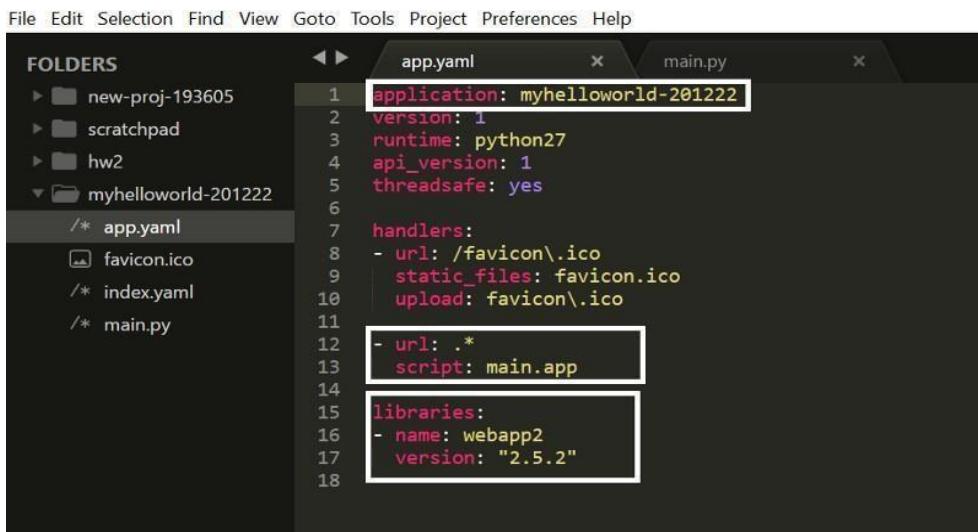
This is the project ID which you should never change. This is the unique identifier for the app

2. url -> script

This is the homepage for the app. In other words, this file will be rendered in your browser when you launch the app

3. libraries

This is where you can include external libraries to use within the webapp



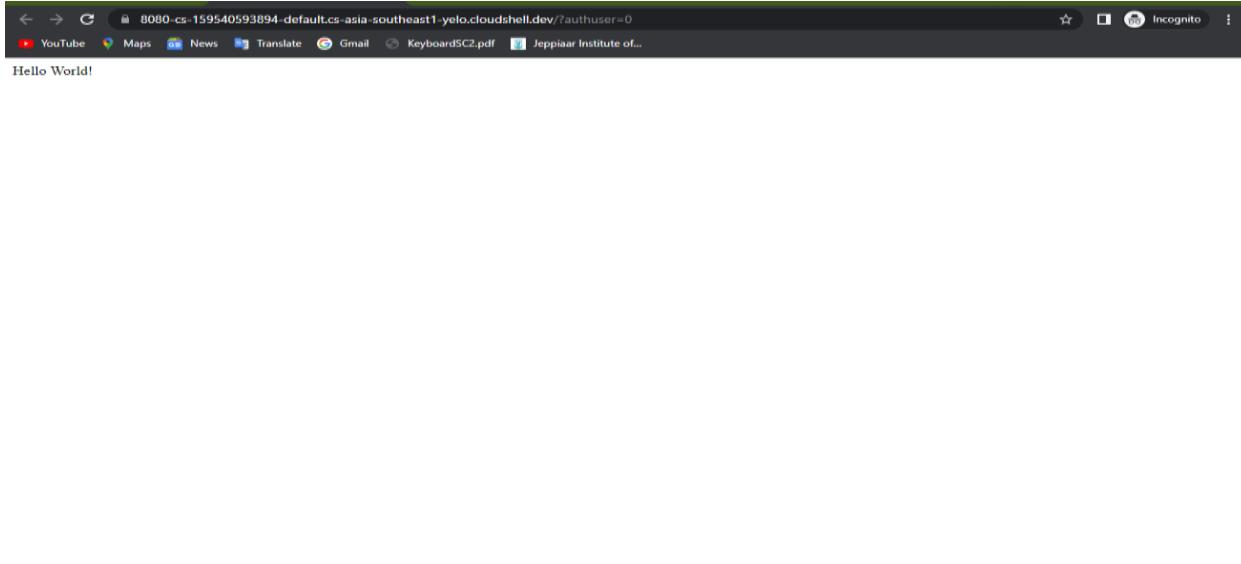
```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS app.yaml main.py
new-proj-193605
scratchpad
hw2
myhelloworld-201222
/* app.yaml
 favicon.ico
/* index.yaml
/* main.py

1 application: myhelloworld-201222
2 version: 1
3 runtime: python27
4 api_version: 1
5 threadsafe: yes
6
7 handlers:
8 - url: /favicon\.ico
9   static_files: favicon.ico
10  upload: favicon\.ico
11
12 - url: .*
13   script: main.app
14
15 libraries:
16 - name: webapp2
17   version: "2.5.2"
```

app.yaml file in the webapp folder

main.py

This is the homepage of the app



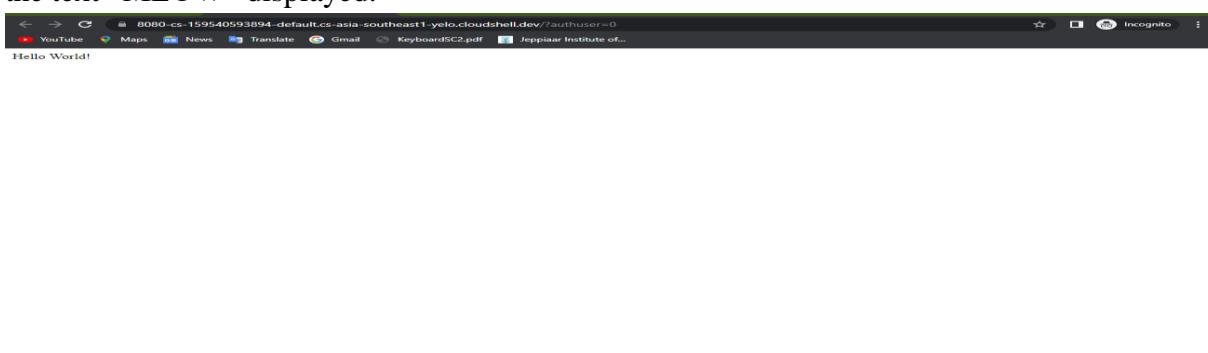
Step 7. Make your changes and deploy the new app

No hello world app is ever complete without the developer changing the hello world text to something else just to make sure that everything happening behind the scenes is working as it should.

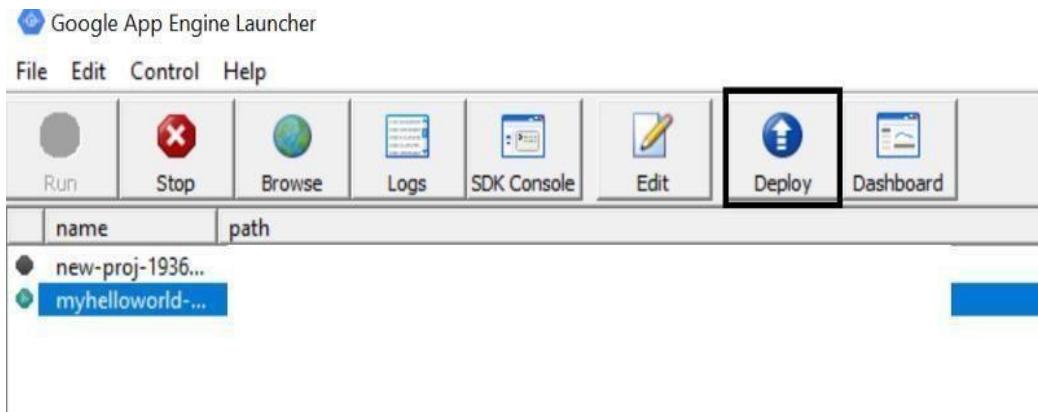
Go ahead and change the text in the above screenshot to something else

```
17 import webapp2
18
19 class MainHandler(webapp2.RequestHandler):
20     def get(self):
21         self.response.write('MEOW')
22
23 app = webapp2.WSGIApplication([
24     ('/', MainHandler)
25 ], debug=True)
26
```

Save the changes, go to the browser and refresh the page. You should see the page with the text “MEOW” displayed.



Finally, it is time to deploy your changes to the cloud to make them globally accessible via a URL. Go to the App Engine launcher, select the app, and hit the **Deploy** button.



go to the URL below:

<https://<yourProjectID>.appspot.com/>

<https://myhelloworld-201222.appspot.com/>



Result :

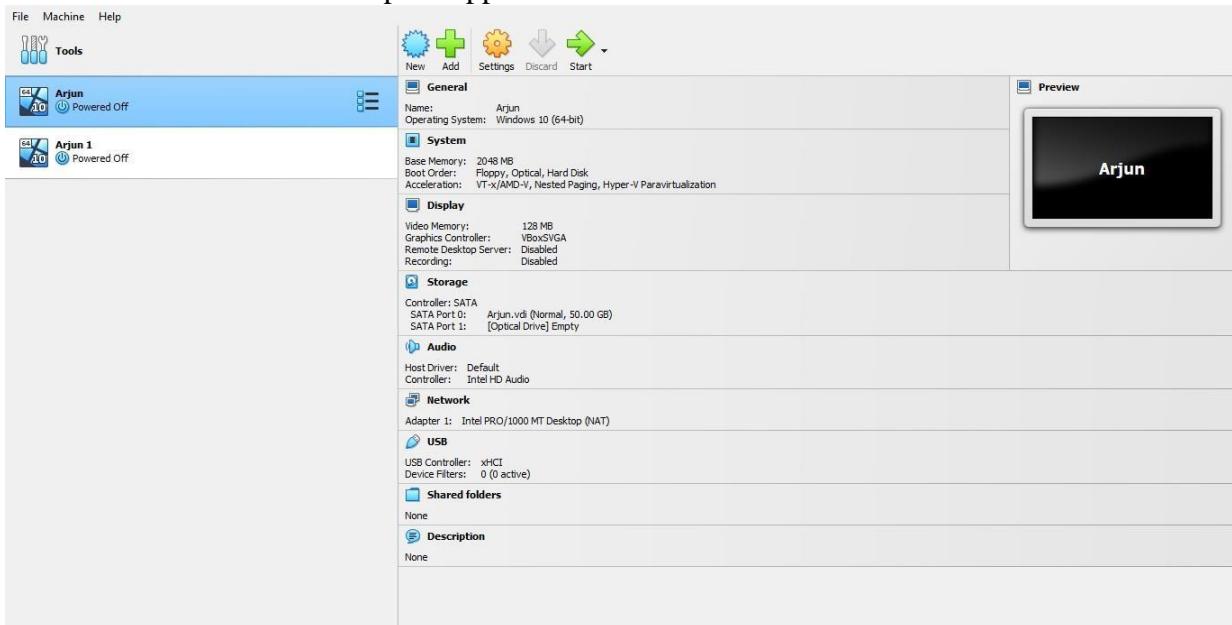
The procedure for GAE launcher to launch the hello world web application using python was successfully learned and verified .

AIM:

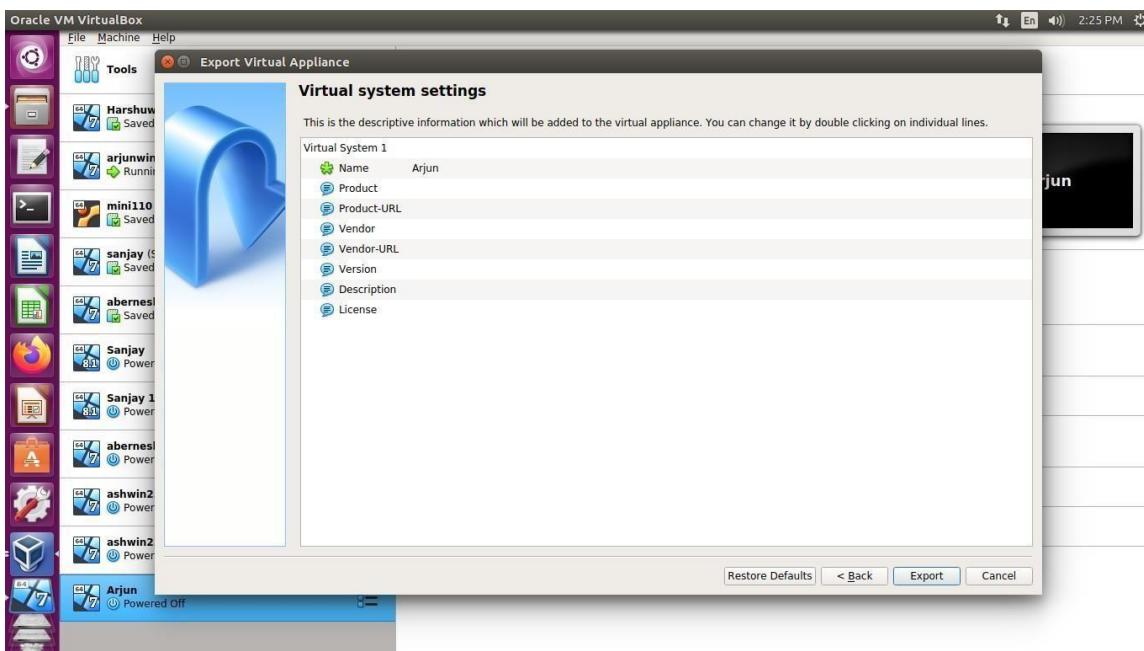
To find a procedure to transfer the files from one virtual machine to another virtual machine by migration based on the certain condition from one node to the other.

PROCEDURE:

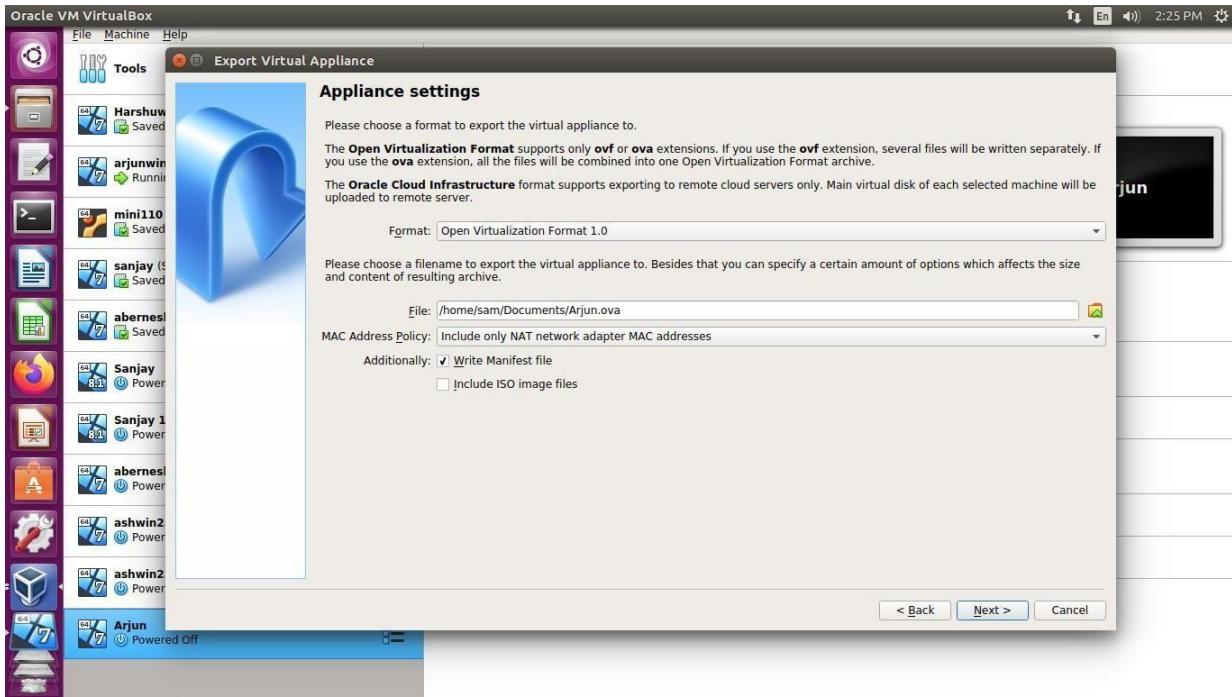
1. Select the VM and click File->Export Appliance



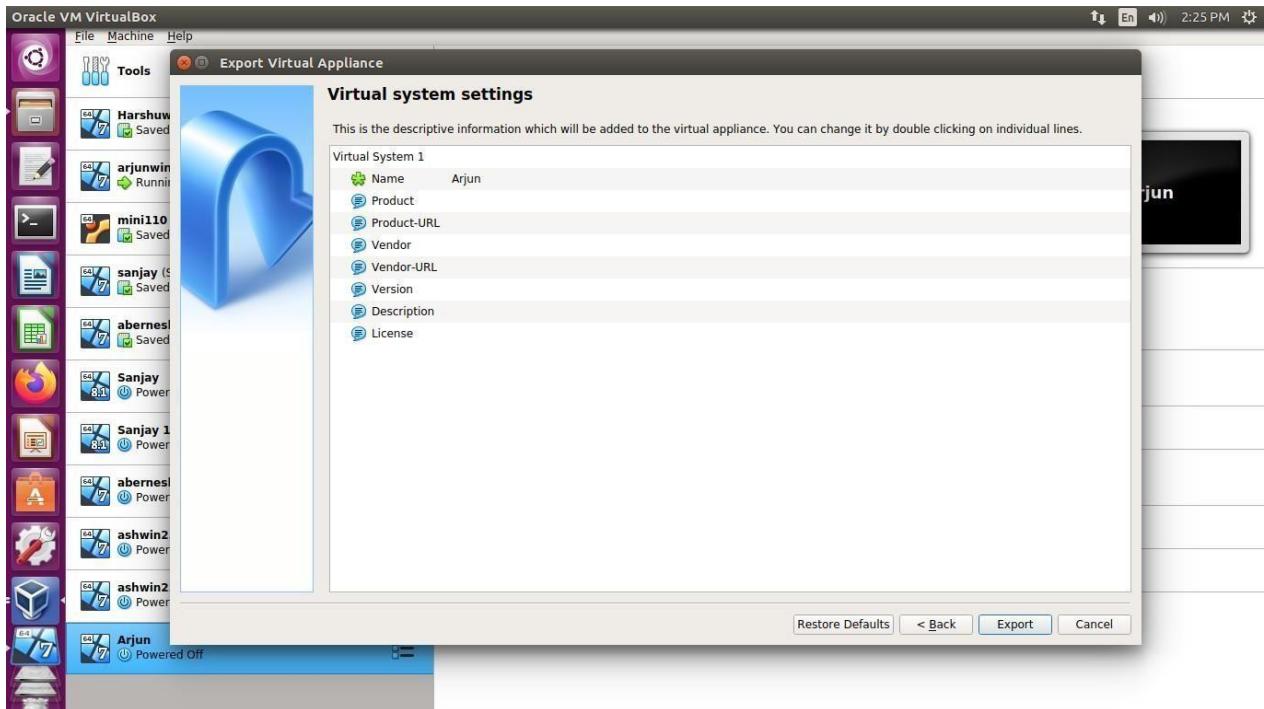
2. Select the VM to be exported and click NEXT.



3. Note the file path and click “Next”

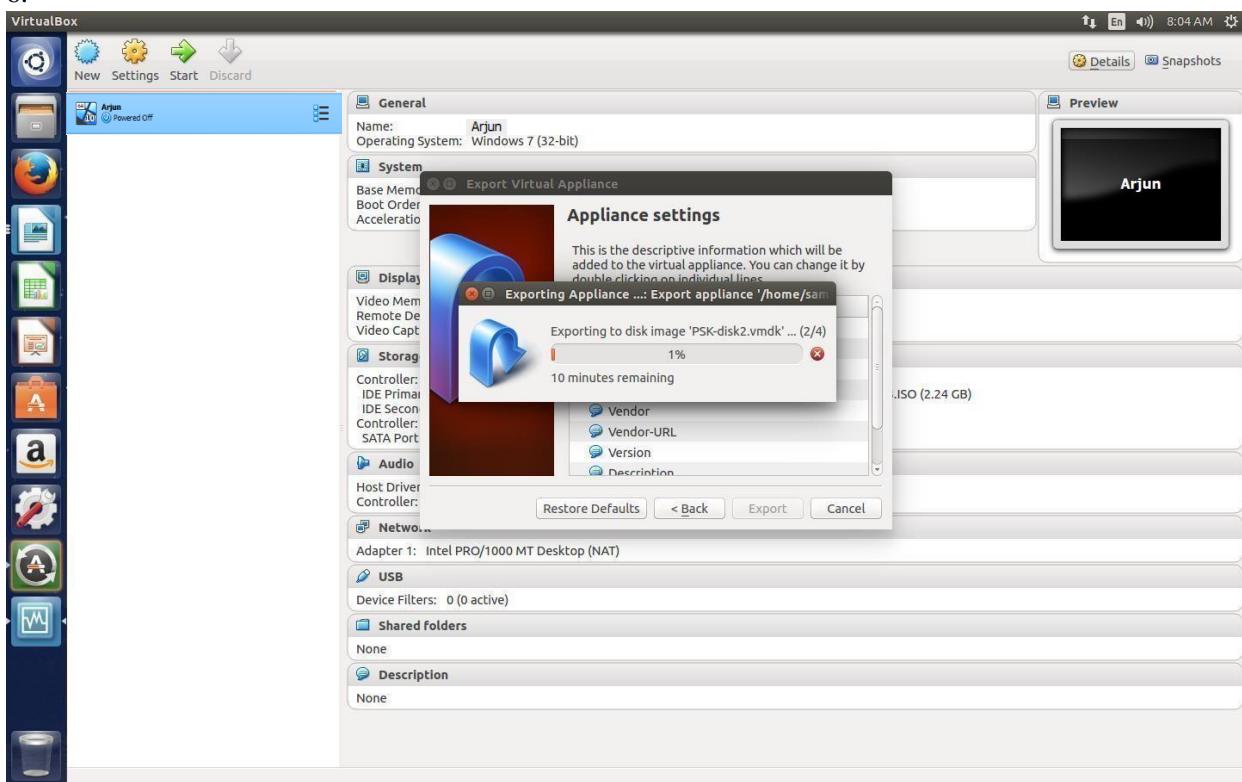


4. Click “Export”

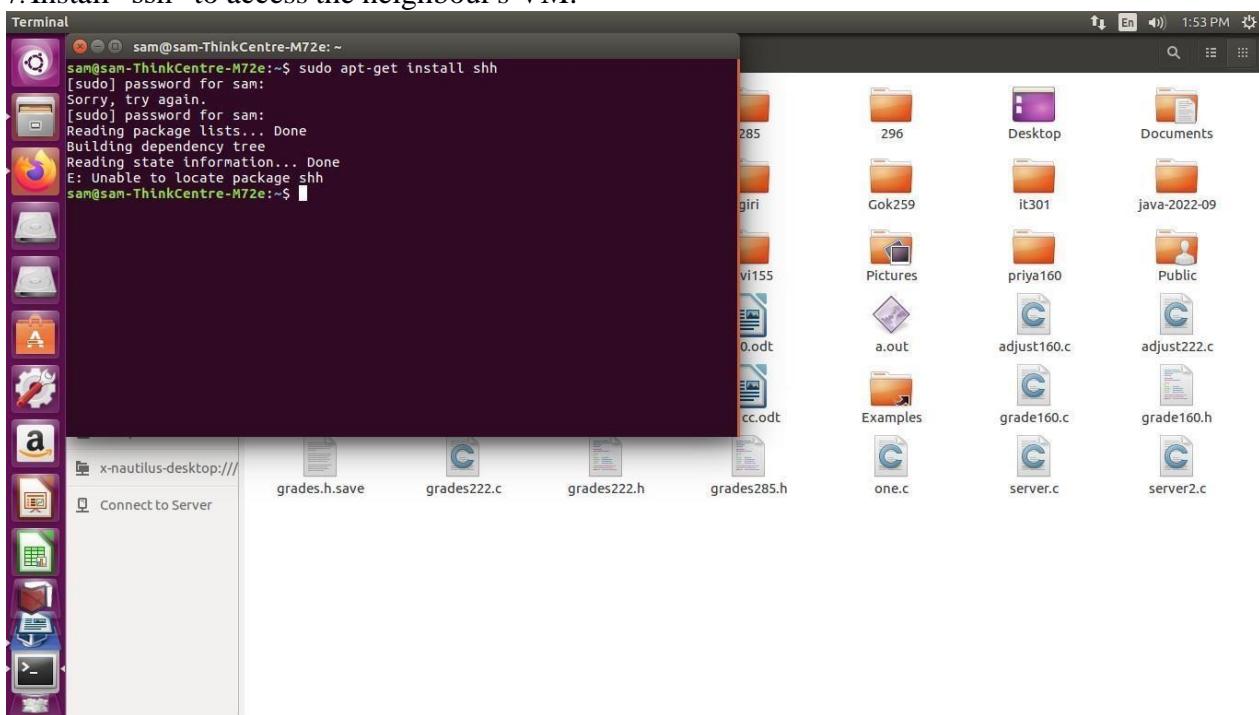


5. The Virtual machine is being exported.

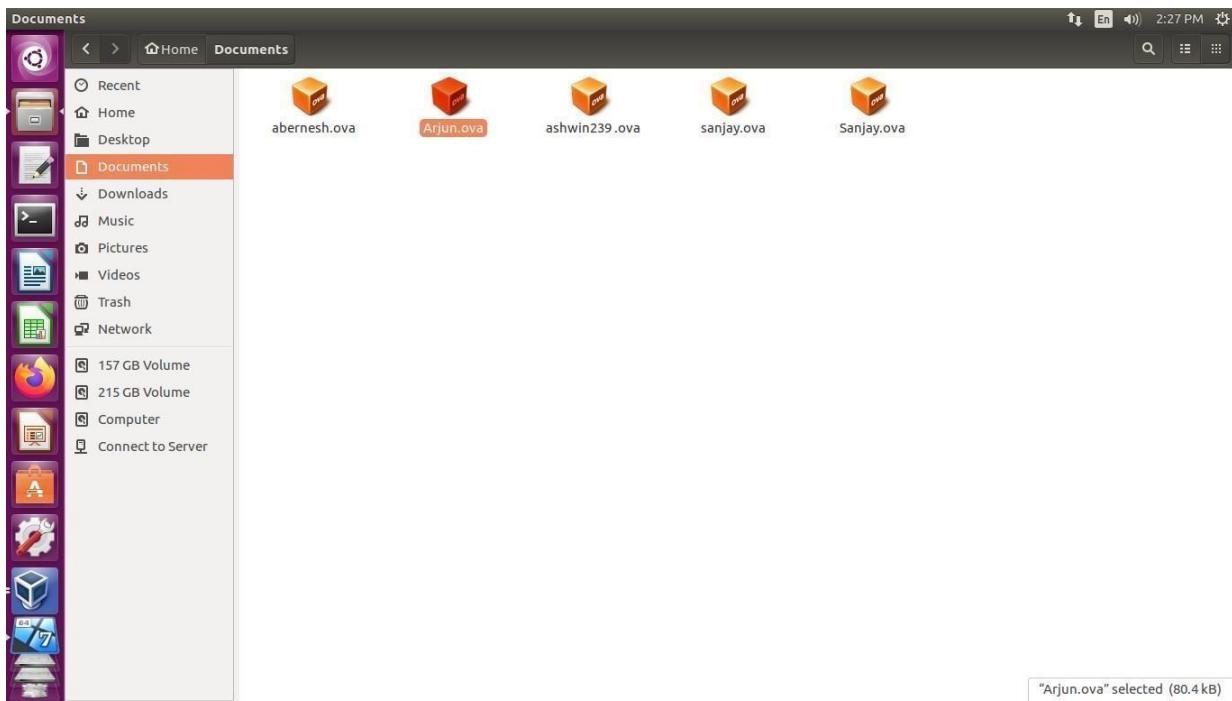
6.



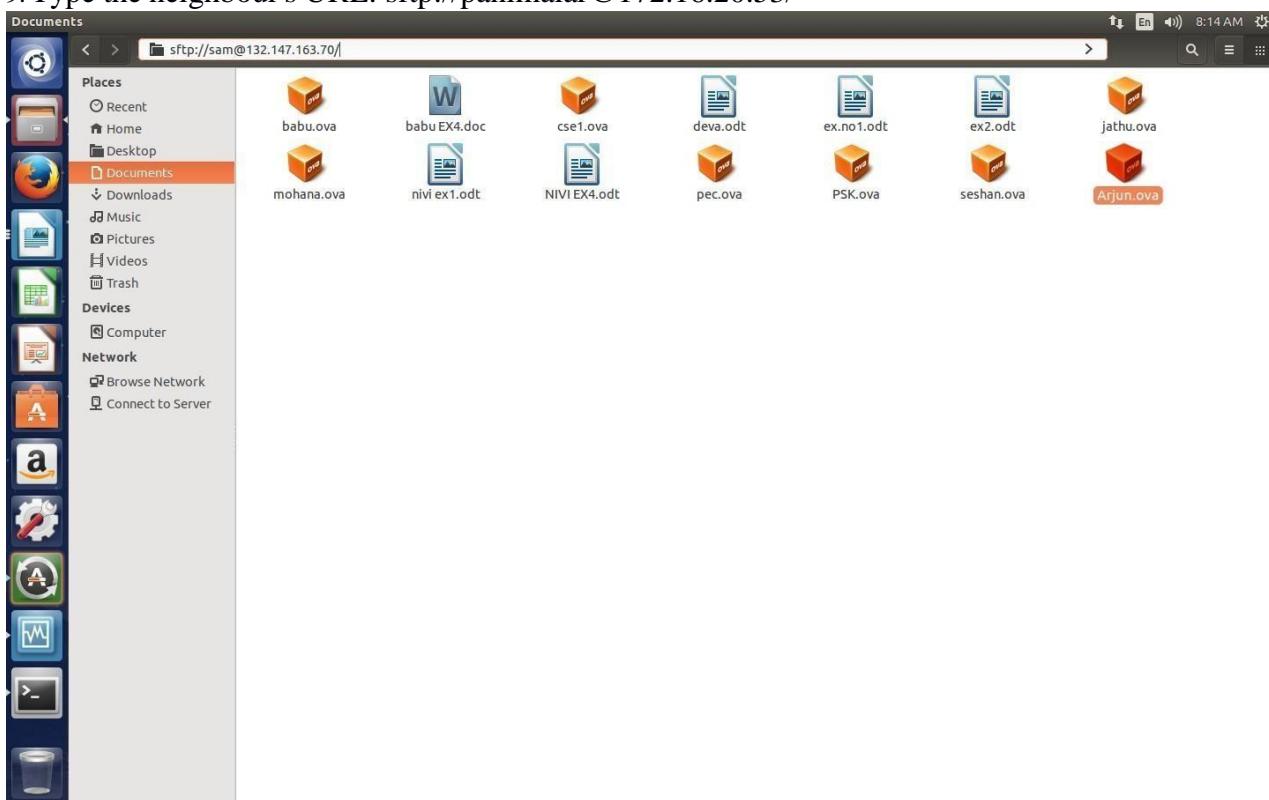
7. Install “ssh” to access the neighbour's VM.



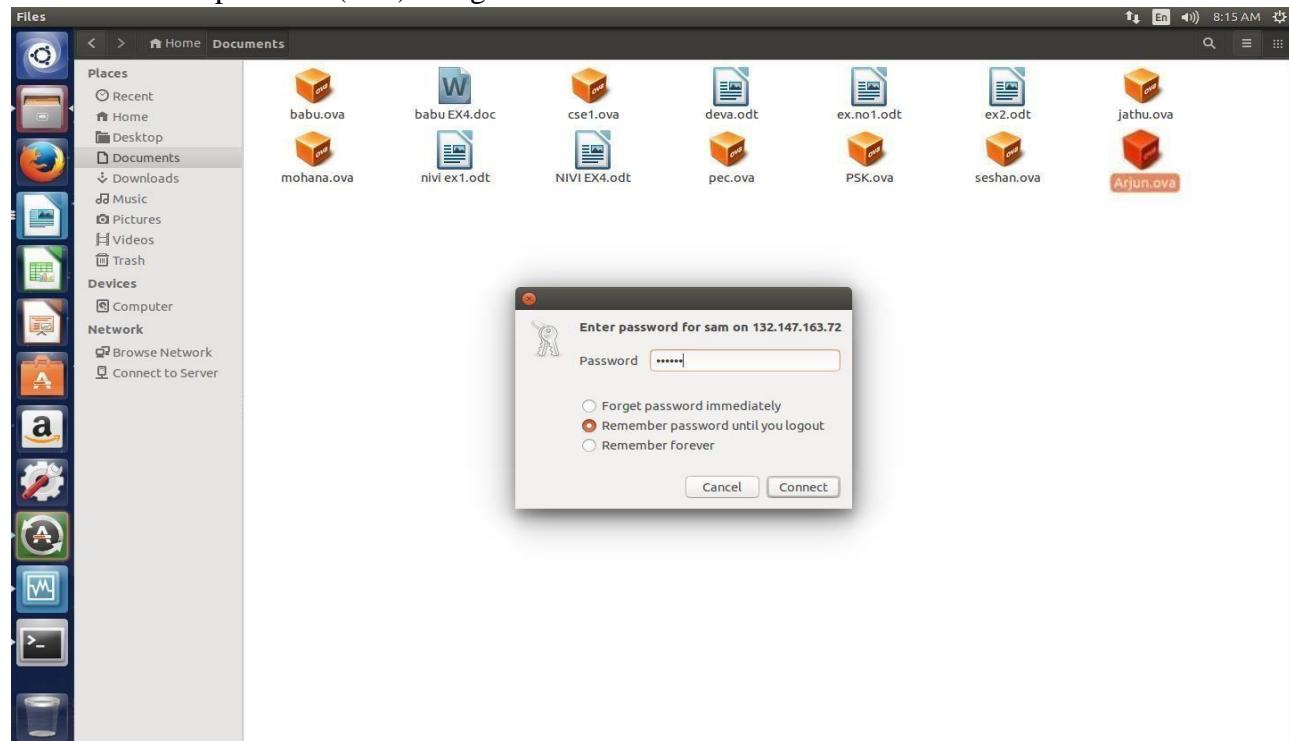
8. Go to File->Computer:/home/panimalar/Documents/



9. Type the neighbour's URL: sftp://panimalar@172.16.20.53/



10. Give the password(root) and get connected.

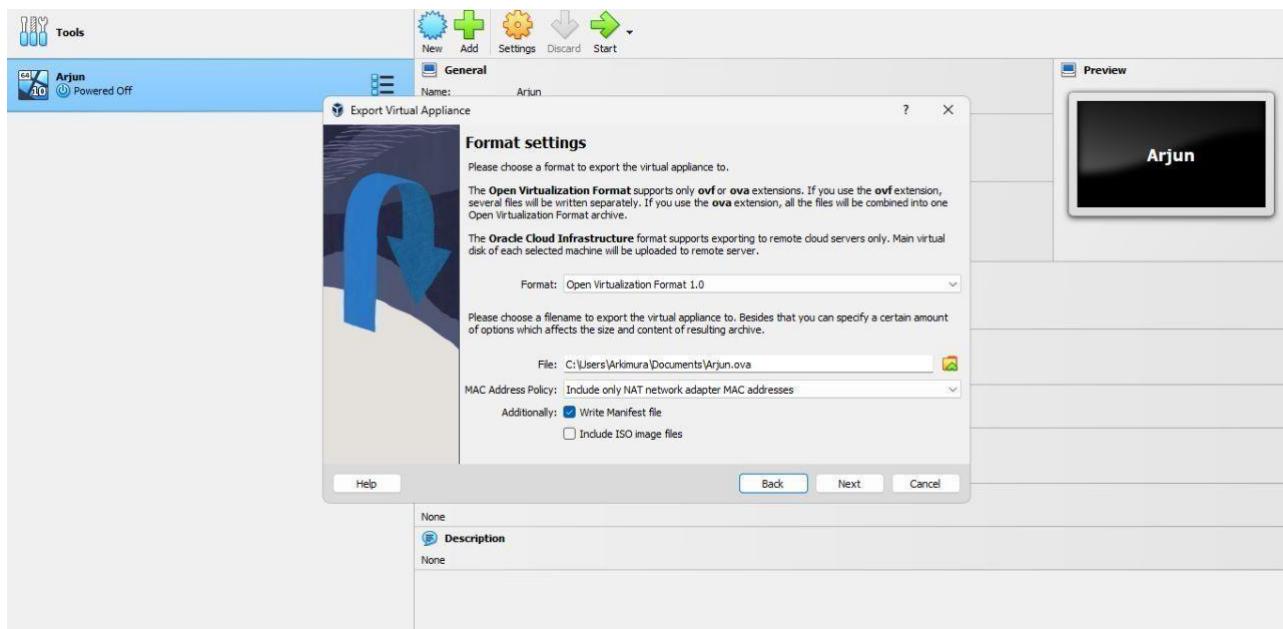


11. Select the VM and copy it in desktop.

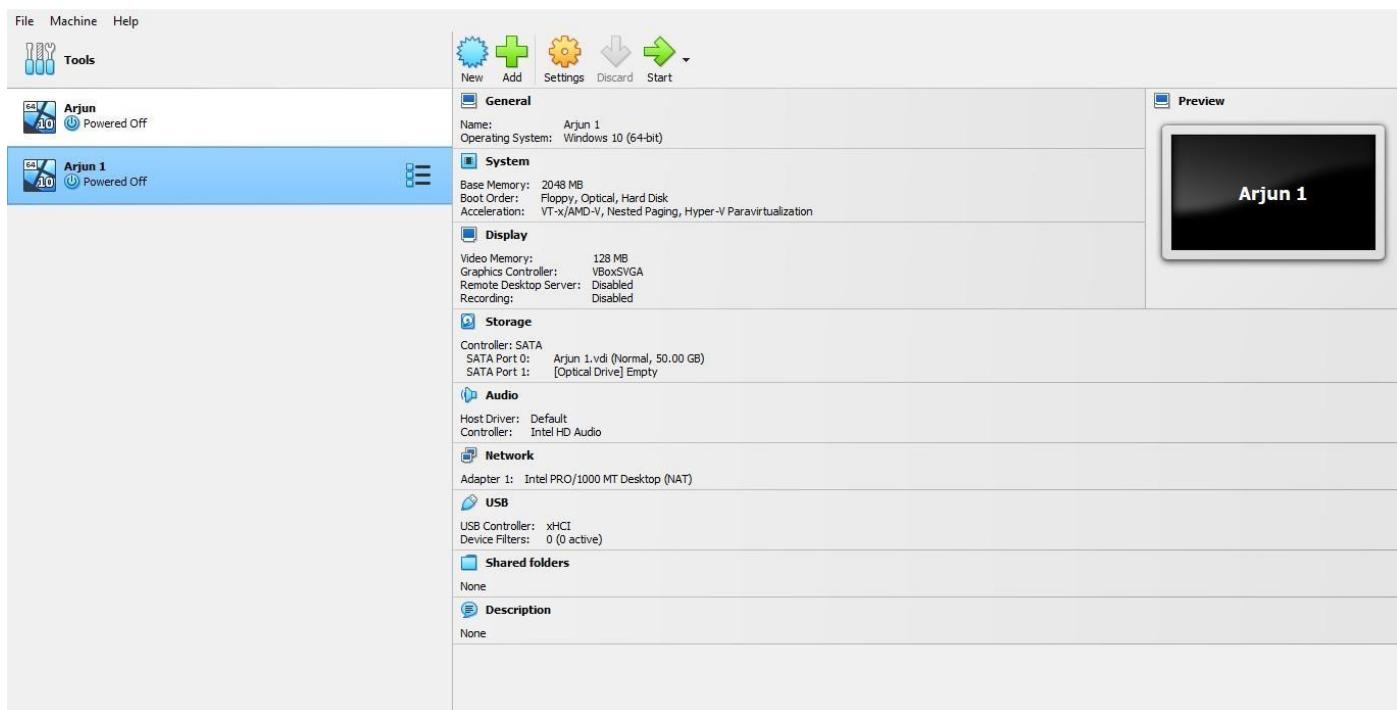
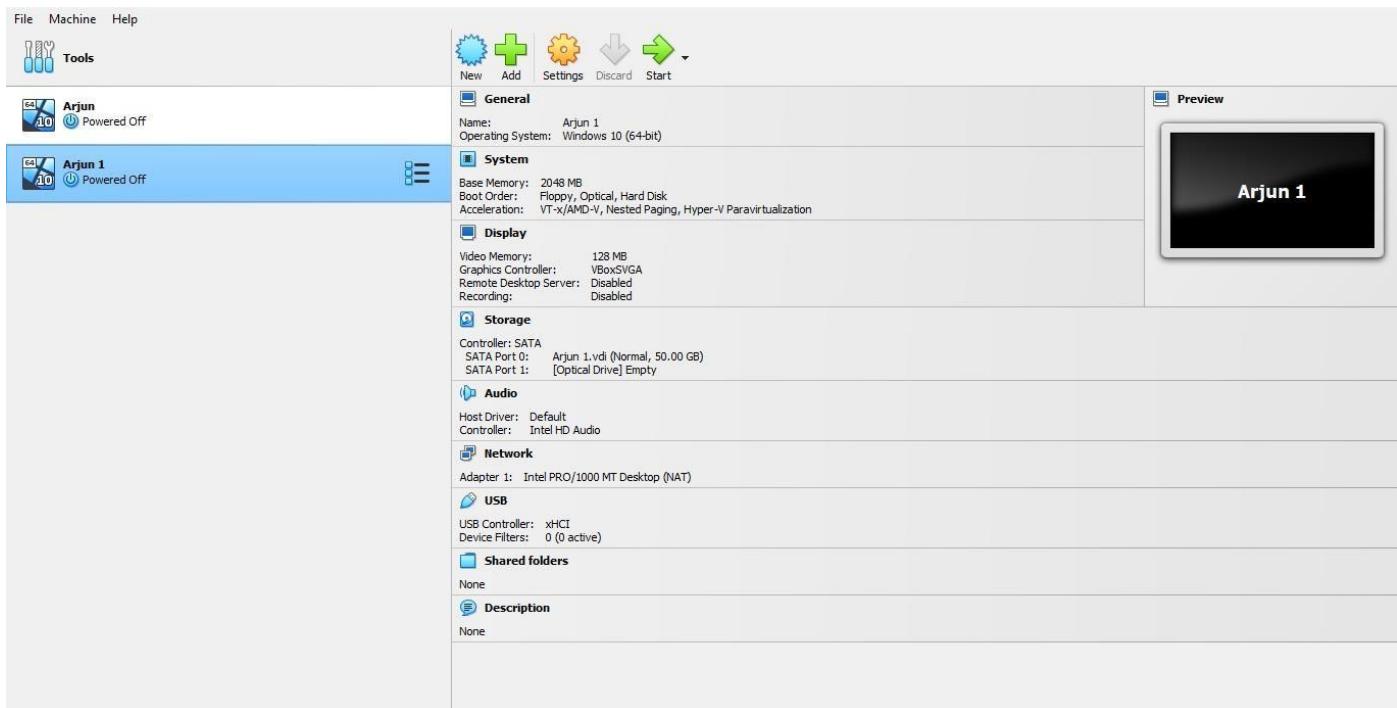
12. Open VirtualBox and select File->Import Appliance->Browse

13. Select the VM to be imported and click "Open".

14. Click "Next"



15. VM is being imported.



16. VM is imported.

RESULT:

The procedure to transfer the files from one virtual machine to another virtual machine by migration based on the certain condition from one node to the other was successfully learned and verified .

Aim :

To find a procedure to launch virtual machine using trystack (online openstack demo version).

Procedure :

OpenStack is an open-source software cloud computing platform. OpenStack is primarily used for deploying an infrastructure as a service (IaaS) solution like Amazon Web Service (AWS).

TryStack is the easiest and free way to do it.

Minimum requirements for OpenStack is listed below:

- 4 GB Of Ram.
- 4 CPU Units.
- 30 GB Disk Space.

Step 1: Prepare the environment for installing OpenStack:

Run the following commands and you will be done with it to **bring all your packages to the latest version and install git so that we can clone OpenStack** to our Linux machine.

Commands :

- › sudo apt-get update
- › sudo apt-get upgrade
- › sudo apt-get dist-upgrade
- › sudo apt-get install git -y
- › sudo reboot

Step 2: Download and Install OpenStack!

Note: If you already have a “stack” user on your virtual machine or laptop (with sudo privileges), then you do not need to create an additional user.

After your virtual machine is done with a reboot, you are now ready to install OpenStack.

Normally **OpenStack runs under non-root user with sudo privileges**. We can easily create one to start with using:

Create the user named as “stack”

sudo useradd -s /bin/bash -d /opt/stack -m stack

Now let us give this user sudo privileges using:

```
echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

We now have to log in as user “stack” to proceed with our installation as

```
sudo su – stack
```

Start with the installation of openstack by downloading the required material.

```
git clone https://git.openstack.org/openstack-dev/devstack
```

```
# cd to the cloned directory
```

```
cd devstack
```

Normally during installing it will ask you to set various passwords, you can automate this process by creating a file in your current directory named “**local.conf**”. Save and exit the following file, this will automate the installation process.

```
# create the file
```

```
$ nano local.conf
```

```
# Now paste following contents in the file
```

```
[[local|localrc]]
```

```
ADMIN_PASSWORD=secret
```

```
DATABASE_PASSWORD=$ADMIN_PASSWORD
```

```
RABBIT_PASSWORD=$ADMIN_PASSWORD
```

```
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

We are now ready to run the installation script. **Installation script can be launched** using the command:

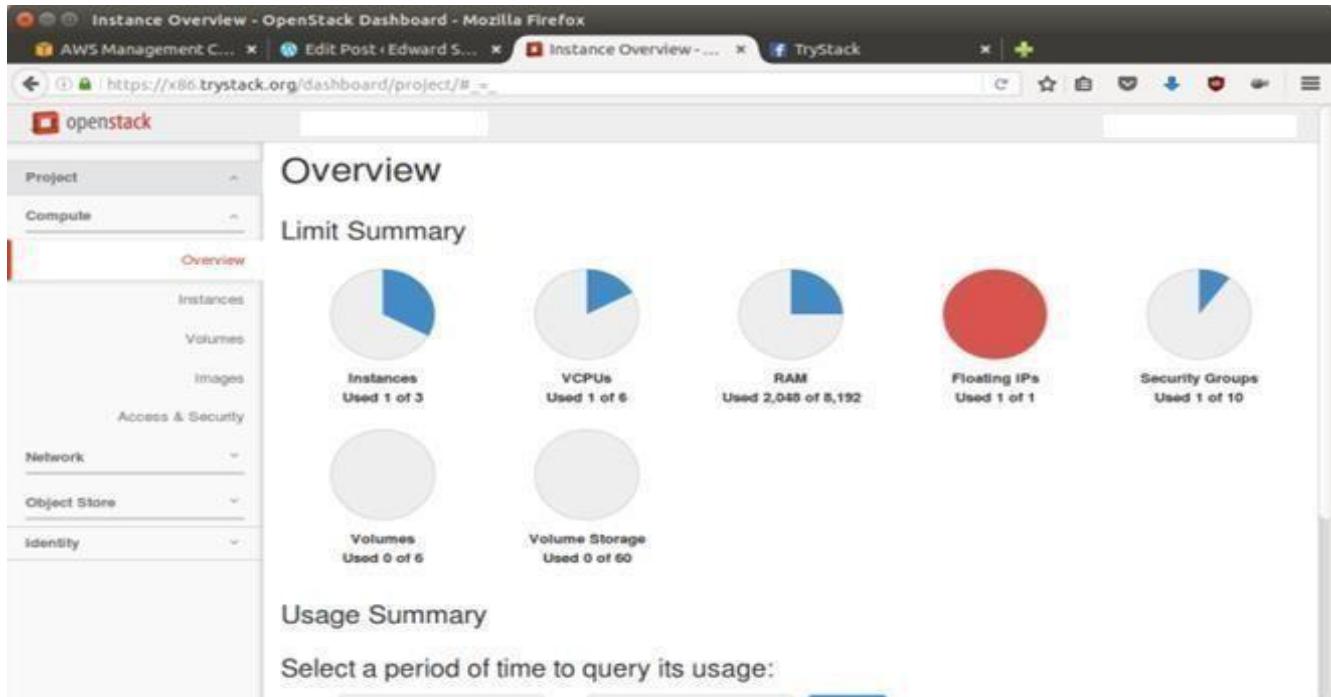
```
./stack.sh
```

Once installation is complete your screen should be :

```
This is your host IP address: [REDACTED]
This is your host IPv6 address: [REDACTED]
Horizon is now available at http://[REDACTED]/dashboard
Keystone is serving at http://[REDACTED]/identity/
The default users are: admin and demo
The password: secret
Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/developer/devstack/systemd.html
2017-06-19 16:13:18.557 | WARNING:
2017-06-19 16:13:18.557 | Using lib/neutron-legacy is deprecated, and it will be removed in the future
2017-06-19 16:13:18.557 | stack.sh completed in 1933 seconds.
stack@devstack:[-]devstack$
```

OpenStack dashboard can now be accessed at :

<http://<IPAddress>/dashboard/>



Step 3: Create Network!

Note: Please remember that without having a network, you can not launch an instance/virtual machine.

After you are logged into the OpenStack Dashboard it will look something like this:

The screenshot shows the 'Projects' page of the OpenStack Dashboard. The top navigation bar shows 'openstack' and 'all_demo *'. The user 'admin' is logged in. The left sidebar has dropdown menus for 'Project', 'Admin', and 'Identity'. The 'Identity' menu is expanded, showing 'Projects'. The main content area has a 'Projects' tab selected. It displays a table with two rows of data. The columns are 'Name', 'Description', 'Project ID', 'Domain Name', 'Enabled', and 'Actions'. The first row has a checkbox, the name 'all_demo', a blank description, the Project ID '04fe70225674c000a1a1fb0f7cd7e4', 'Default' domain, and 'Yes' enabled status. The second row has a checkbox, the name 'demo', a blank description, the Project ID '32facaa7b3461000728801111d988', 'Default' domain, and 'Yes' enabled status. There are 'Create Project' and 'Delete Project' buttons at the bottom of the table.

We need to **create a network that virtual machine can use**. For now, it will just be a dummy network:

Steps :

1. Click the Project Drop Down.
2. Click the Network Drop Down.
3. From network Drop Down select Networks, and this window will open that you see on the right side.
4. Finally, click Create Network.

The screenshot shows the 'Networks' section of a cloud provider's management console. The left sidebar has sections for Project, API Access, Compute, Volumes, Network (selected), Network Topology, Routers, Security Groups, Floating IPs, Admin, and Identity. The main area is titled 'Networks' and shows one item: 'public'. A red arrow labeled '1' points to the 'Project' dropdown in the top navigation bar. A red arrow labeled '2' points to the 'Network' dropdown in the sidebar. A red arrow labeled '3' points to the 'Networks' tab in the sidebar. A red arrow labeled '4' points to the 'Create Network' button at the top right of the main table area.

| Name | Subnets Associated | Shared | External | Status | Admin State | Actions |
|--------|---|--------|----------|--------|-------------|-------------------------------|
| public | Ipv6-public-subnet 2001:db8::/64 public-subnet 172.24.0.0/24 | No | Yes | Active | UP | <button>Edit Network</button> |

The screenshot shows the 'Create Network' wizard. The title bar says 'Create Network'. The tabs at the top are 'Network' (selected), 'Subnet', and 'Subnet Details'.
Network Name: CyberPersons
Description: Create a new network. In addition, a subnet associated with the network can be created in the following steps of this wizard.
Enable Admin State: (with a tooltip)
Shared:
Create Subnet:
Buttons: Cancel, Back, Next >

Create Network

Network Subnet Subnet Details

1 Subnet Name
CyberPersons

2 Network Address Source
Allocate Network Address from a pool

3 Address pool
shared-default-subnetpool (10.0.0.0/22)

4 Network Mask
26 (pool default)

5 IP Version
IPv4

Gateway IP ?

Cancel « Back Next »

Creates a subnet associated with the network. You need to enter a valid "Network Address" and "Gateway IP". If you did not enter the "Gateway IP", the first value of a network will be assigned by default. If you do not want gateway please check the "Disable Gateway" checkbox. Advanced configuration is available by clicking on the "Subnet Details" tab.

Once all these things are done, click “Next”.

Now everything is optional in this below window if you are interested in filling something up, you can. Otherwise, leave everything as it is and click “Create”. You now have a network that you can use to launch a virtual machine.

Create Network

Network Subnet Subnet Details

Enable DHCP

Allocation Pools ?

DNS Name Servers ?

Host Routes ?

Cancel « Back Create

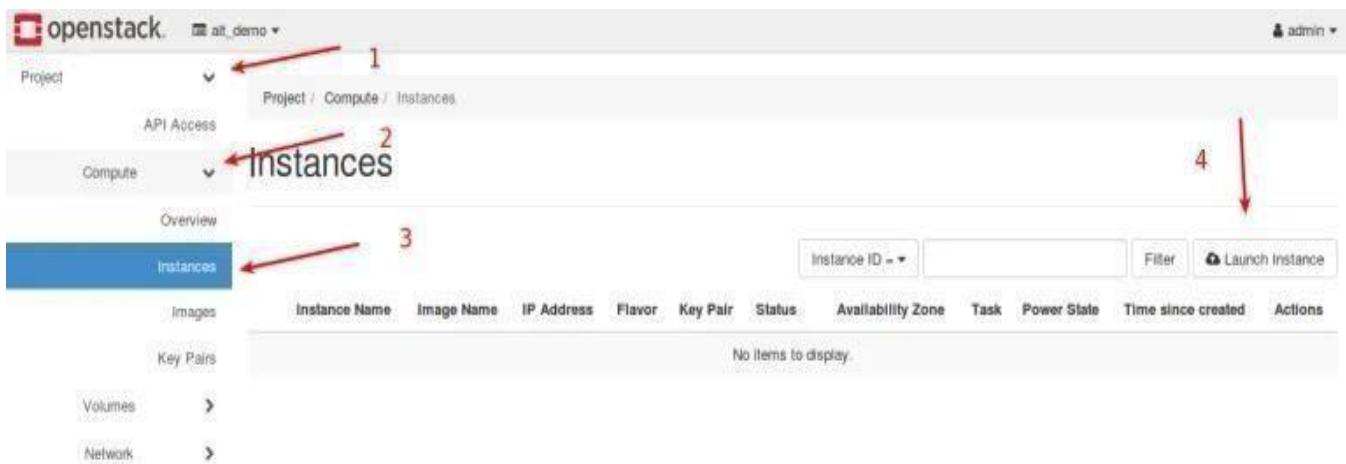
Specify additional attributes for the subnet.

Step 4: Create Virtual Machine/Instance

After the network is created, we are now ready to create our very first virtual machine.

Steps :

1. Click on “Project” drop down.
2. Inside project click “Compute” drop down.
3. Under compute you have four options, since we are interested in creating an instance, you have to click on “Instance”.
4. Finally, click “Launch Instance”.



Launch Instance

Details

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name *

Flavor *

Availability Zone

Count *

Total Instances (10 Max)
10%
0 Current Usage
1 Added
9 Remaining

Source *

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Now, there are 11 tabs to create an instance, we will go through each tab one by one.

Details Tab :

This is a general information tab for creating an instance,

- You will have to **assign a name to your virtual machine** on this tab.
- Select **zone to launch a virtual machine**, and
- Tell **how many copies of virtual machine** you want.

Launch Instance

Details

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name * CyberPersons

Availability Zone nova

Total Instances (10 Max) 10%

Count * 1

Source *

Flavor *

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Cancel Back Next Launch Instance

Source Tab

Normally when we create a virtual machine on Proxmox or VMWare we need to insert CD-ROM

In OpenStack this is done by Source Tab, you can use various ways to launch a new virtual machine, OpenStack allows you to choose **following as a source to create your instance**.

- **Image**
- **Snapshot of already created instance**
- **Volume or a volume Snapshot**

We are going to use “Cirros” image to create our instance.

Launch Instance

Source *

| Select Boot Source | | Create New Volume |
|--------------------|----------------------------------|-------------------|
| Image | Yes | No |
| Volume Size (GB) * | Delete Volume on Instance Delete | |
| 1 | Yes | No |

Security Groups

| Name | Updated | Size | Type | Visibility |
|---|-----------------|-----------|-------|------------|
| Select an item from Available items below | | | | |
| Available <small>(2)</small> | | | | |
| Click here for filters. | | | | |
| ubuntu | 6/22/17 3:17 AM | 829.00 MB | iso | Public |
| cirros-0.3.5-x86_64-disk | 6/22/17 2:14 AM | 12.65 MB | qcow2 | Public |

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Cancel **Next >** **Launch Instance**

1. Click on the icon where the first arrow is pointing, so that we can use “Cirros” to launch our virtual machine.
2. After the image is selected, just click “Next” so that we can move to “Flavor” tab.

Flavor Tab

Flavor tab will **allow you to allocate resource to your instance**. Like:

- Ram.
- CPU.
- Disk Space.

It is similar to giving virtual resources to the virtual machine, but OpenStack gives fancy names to everything

The screenshot shows the 'Launch Instance' wizard. The 'Flavor' tab is selected. On the left, there's a sidebar with options like Details, Source, Networks, Network Ports, Security Groups, Key Pair, Configuration (which has a red arrow pointing to it), Server Groups, Scheduler Hints, and Metadata. The main area shows two tables: 'Allocated' (with one entry for m1.tiny) and 'Available' (with 11 entries: m1.small, m1.medium, m1.large, m1.nano, m1.xlarge). At the bottom are 'Cancel', 'Back', 'Next', and 'Launch Instance' buttons.

| Name | VCPUS | RAM | Total Disk | Root Disk | Ephemeral Disk | Public |
|---------|-------|--------|------------|-----------|----------------|--------|
| m1.tiny | 1 | 512 MB | 1 GB | 1 GB | 0 GB | Yes |

| Name | VCPUS | RAM | Total Disk | Root Disk | Ephemeral Disk | Public |
|-----------|-------|-------|------------|-----------|----------------|--------|
| m1.small | 1 | 2 GB | 20 GB | 20 GB | 0 GB | Yes |
| m1.medium | 2 | 4 GB | 40 GB | 40 GB | 0 GB | Yes |
| m1.large | 4 | 8 GB | 80 GB | 80 GB | 0 GB | Yes |
| m1.nano | 1 | 64 MB | 0 GB | 0 GB | 0 GB | Yes |
| m1.xlarge | 8 | 16 GB | 160 GB | 160 GB | 0 GB | Yes |

You can see that there are 11 available pre-configured templates to choose from. The one I choose gave following resources to the instance:

- **1 virtual CPU.**
- **512 MB Ram.**
- **1 GB Disk.**

After flavor is selected, just press “Next”.

Network Tab

Network tab **allows us to define a network for our virtual machine**, you might have remembered that we've created a network above for this purpose.

Now by default, the network you have created above will be selected for this machine, as seen in the image below:

The screenshot shows the 'Launch Instance' dialog box with the 'Networks' tab selected. The 'Allocated' section displays one network named 'CyberPersons' with its status as 'Up' and 'Active'. The 'Available' section is empty with a note to select at least one network. Other tabs like 'Details', 'Source', 'Flavor', 'Security Groups', 'Key Pair', and 'Configuration' are also visible.

Don't change anything just click "Next".

Network Ports Tab

For now, just leave the default settings on "Network Ports" tab and click next.

Security Groups Tab

Security groups define how a specific virtual machine is allowed to talk with the outer world. As for now, we are just trying to create our first virtual machine, you can leave all the defaults.

Key-Pair Tab

Leave defaults and click Next.

Configuration Tab

Leave defaults and click Next.

Server Groups Tab

Leave defaults and click Next.

Scheduler Hints Tab

Leave defaults and click Next.

Metadata Tab

Leave defaults and click Next.

Launch Instance

After going through all the tabs, you are now ready to press that magic "Launch Instance" button.

Once you click "Launch Instance" button, OpenStack will start creating our virtual machine, and it is going to look something like this:

Displaying 1 item

| Instance Name | Image Name | IP Address | Flavor | Key Pair | Status | Availability Zone | Task | Power State | Time since created | Actions |
|---------------|------------|------------|---------|----------|--------|-------------------|----------------------|-------------|--------------------|---|
| CyberPersons | - | 10.0.0.75 | m1.tiny | - | Build | nova | Block Device Mapping | No State | 0 minutes | Associate Floating IP ▾ |

Displaying 1 item

Step 5: Access Virtual Machine Console!

Once you click “Launch Instance” it will take OpenStack few seconds to create your virtual machine. Once ready you can access the console to see how the command line of your first virtual machine looks like.

Displaying 1 item

| Instance Name | Image Name | IP Address | Flavor | Key Pair | Status | Availability Zone | Task | Power State | Time since created | Actions |
|---------------|------------|----------------------------|---------|----------|--------|-------------------|------|-------------|--------------------|-----------------------------------|
| CyberPersons | - | 172.24.4.11 2001:db8::6 | m1.tiny | - | Active | nova | None | Running | 3 minutes | Create Snapshot ▾ |

Displaying 1 item



- [Associate Floating IP](#)
- [Attach Interface](#)
- [Detach Interface](#)
- [Edit Instance](#)
- [Attach Volume](#)
- [Detach Volume](#)
- [Update Metadata](#)
- [Edit Security Groups](#)
- [Console](#)
- [View Log](#)
- [Pause Instance](#)
- [Suspend Instance](#)
- [Shelve Instance](#)
- [Resize Instance](#)
- [Lock Instance](#)
- [Soft Reboot Instance](#)
- [Hard Reboot Instance](#)
- [Shut Off Instance](#)

Click on “Console” and OpenStack will take you to the console of the virtual machine. The console will look something like this:

```
Connected (unencrypted) to: QEMU (Instance-00000001) Send C
[ 2.164811] NET: Registered protocol family 10
[ 2.180216] NET: Registered protocol family 17
[ 2.181637] Registering the dns_resolver key type
[ 2.221567] registered taskstats version 1
[ 2.251264] Freeing initrd memory: 3452k freed
[ 2.406290] Magic number: 5:833:342
[ 2.407395] rtc_cmos 00:01: setting system clock to 2017-06-22 18:18:41 UTC (1498155521)
[ 2.410548] powernow-k8: Processor cpuid 663 not supported
[ 2.414248] BIOS EDD facility v0.16 2004-Jun-25, 0 devices found
[ 2.414675] EDD information not available.
[ 2.440477] Freeing unused kernel memory: 928k freed
[ 2.469513] Write protecting the kernel read-only data: 12288k
[ 2.521076] Freeing unused kernel memory: 1596k freed
[ 2.560647] Freeing unused kernel memory: 1184k freed
[ 2.563628] Refined TSC clocksource calibration: 2593.959 MHz.
[ 2.564469] Switching to clocksource tsc

further output written to /dev/ttys0
```

Result :

The procedure for find a procedure to launch virtual machine using trystack (online openstack demo version) was learned and verified successfully .

Aim:

To find procedure to set up the one node Hadoop cluster using Hadoop Apache open source framework with simple programming models .

Procedure :

1. Start and complete the perquisite procedure .

2. Creating namenode and datanode and configuring:

- a. hduser@sysk38\$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode**
- b. hduser@sysk38\$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode**
- c. hduser@sysk38\$ **sudo chown -R hduser:hadoop /usr/local/hadoop_store**
- d. hduser@sysk38\$ **sudo chown -R hduser /usr/local/hadoop**

3. Formatting namenode as preliminary stage of HDFS:

hduser@sysk38\$ **/usr/local/hadoop/bin/hadoop namenode -format**

4. Starting Hadoop:

hduser@sysk38\$ **/usr/local/hadoop/sbin/start-all.sh**

5. Checking Hadoop:

hduser@sysk38\$ **jps**

6. Checking HDFS installing:

\$ hadoop version

7. Creating HDFS Input Directory:

\$hadoop fs -mkdir /usr/local/hadoop/input

8. Copying a file into HDFS Input Directory:

\$ hadoop fs -put /home/4300.txt /usr/local/hadoop/input

9. Stoping Hadoop:

hduser@sysk38\$ **stop-all.sh**

Commands and Procedure :

I. Installing JAVA JDK

Login as root:

1. sudo -s

Install Jave Runtime :

2. sudo apt-get update

3. sudo apt-get install default-jre

Install Java JDK (OpenJDK v 1.7 or newer)

4. sudo apt-get update

5. sudo apt-get install default-jdk

Checking Java verison :

6. Java -version

Adding a dedicated Hadoop user :

Adding group:

7. \$ sudo addgroup hadoop

Adding group `hadoop' (GID 1002) ...

Done.

Creating a user and adding the user to a group:

8. \$ sudo adduser --ingroup hadoop hduser

Adding user `hduser' ...

Adding new user `hduser' (1001) with group `hadoop' ...

Creating home directory `/home/hduser' ...

Copying files from `/etc/skel' ...

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

Changing the user information for hduser

Enter the new value, or press ENTER for the default

Full Name []:

Room Number []:

Work Phone []:

Home Phone []:

Other []:

Is the information correct? [Y/n] **Y**

0 Configuring SSH access:

II. Install Secure Shell

Install (SSH and RSYNC)

ssh has two main components:

- **ssh** : The command we use to connect to remote machines - the client.
- **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first.

Use this command to do that :

9. sudo apt-get install ssh

10. sudo apt-get install rsync

we can think it is setup properly:

11. k@laptop:~\$ which ssh

/usr/bin/ssh

12. k@laptop:~\$ which sshd

/usr/sbin/sshd

Create and Setup SSH Certificates (Setup passphraseless ssh)

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost. So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

Generating an SSH key for the hduser user.

a. Login as hduser with sudo

13.k@laptop:~\$ su hduser

Password:

b. Run this Key generation command:

14.k@laptop:~\$ ssh-keygen -t rsa -P ""

Generating public/private rsa key pair.

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

Created directory '/home/hduser/.ssh'.

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:

50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop

The key's randomart image is:

+--[RSA 2048] ---+

| .oo.o |

```
| ..o=.o |
| .+.o. |
| o= E |
| S+ |
| .+ |
| O+ |
| Oo |
| o.. |
+-----+
```

Enable SSH access to your local machine with this newly created key.

```
15. hduser@laptop:/home/k$ cat $HOME/.ssh/id_rsa.pub >>
$HOME/.ssh/authorized_keys
```

The final step is to test the SSH setup by connecting to your local machine with the hduser user.

We can check if ssh works:

```
16. hduser@laptop:/home/k$ ssh localhost
```

The authenticity of host 'localhost (127.0.0.1)' can't be established.

ECDSA key fingerprint is e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.

Are you sure you want to continue connecting (yes/no)? **yes**

Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86_64)

(OR)

To enable password-less login, generate a new SSH key with an empty passphrase:

Use Hadoop User:

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

1 Disabling IPv6.

```
sudo gedit /etc/sysctl.conf
```

Add the following lines to the end of the file and reboot the machine, to update the configurations correctly.

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

```
net.ipv6.conf.lo.disable_ipv6 = 1
```

III. Fetch and Install Hadoop

Run this following command to download Hadoop version 2.6.0

Fetch Hadoop (Stable Version)

```
17.hduser@laptop:~$ wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
```

Unpack the compressed hadoop file by using this command:

```
18.hduser@laptop:~$ tar xvzf hadoop-2.6.0.tar.gz
```

Move hadoop-2.6.0 to hadoop directory by using give command :

```
19.hduser@laptop:~$mv hadoop-2.6.0 /usr/local/hadoop
```

mv: cannot move ‘hadoop-2.6.0’ to ‘/usr/local/hadoop’: Permission denied

```
20.hduser@laptop:~$ cd hadoop-2.6.0
```

Move hadoop package of your choice, I picked **/usr/local** for my convenience :

```
21.hduser@laptop:~/hadoop-2.6.0$ sudo mv * /usr/local/hadoop
```

[sudo] password for hduser:

hduser is not in the sudoers file. This incident will be reported.

This error can be resolved by logging in as a root user, and then add **hduser** to **sudo**:

```
22.hduser@sysk81:~/hadoop-2.6.0$ su panimalar
```

Password:

```
23.panimalar@sysk81:/home/hduser/hadoop-2.6.0$ sudo adduser hduser sudo
```

[sudo] password for k:

Adding user `hduser' to group `sudo' ...

Adding user hduser to group sudo

Done.

Now, the **hduser** has root privilege, we can move the Hadoop installation to the **/usr/local/hadoop** directory without any problem:

```
24.panimalar@sysk81:/home/hduser/hadoop-2.6.0$ sudo su hduser
```

```
hduser@sysk81:~/hadoop-2.6.0$
```

```
25.hduser@sysk81:/home/panimalar$ sudo -s
```

```
26.root@sysk81:/home/panimalar# mkdir /usr/local/hadoop
```

```
27.root@sysk81:/home/panimalar# exit
```

exit

```
28.hduser@sysk81:/home/panimalar$ sudo su hduser
```

[sudo] password for hduser:

```
29.hduser@sysk81:/home/panimalar$ pwd
```

```
/home/panimalar  
30.hduser@sysk81:/home/panimalar$ cd /home/hduser  
31.hduser@sysk81:~$ cd hadoop-2.6.0/  
32.hduser@sysk81:~/hadoop-2.6.0$ sudo mv * /usr/local/hadoop
```

Make sure to change the owner of all the files to the hduser user and hadoop group by using this command:

```
33.hduser@sysk81:~/hadoop-2.6.0$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

IV. Edit Hadoop Configuration Files

Get Java Correct Path

```
34.hduser@laptop$ update-alternatives --config java
```

There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

Nothing to configure.

```
/usr/lib/jvm/java-7-openjdk-i386 /jre/bin/java
```

Setup Configuration Files

The following files will have to be modified to complete the Hadoop setup:

Modifing The following Files

- ✓ ~/.bashrc
- ✓ /usr/local/hadoop/etc/hadoop/hadoop-env.sh
- ✓ /usr/local/hadoop/etc/hadoop/core-site.xml
- ✓ /usr/local/hadoop/etc/hadoop/yarn-site.xml
- ✓ /usr/local/hadoop/etc/hadoop/mapred-site.xml
- ✓ /usr/local/hadoop/etc/hadoop/hdfs-site.xml

(1) ~/.bashrc :

```
35.hduser@sysk81:~$ nano ~/.bashrc
```

Add the following to the end file after modify the java_home and Hadoop_install,

Change JAVA_HOME to Correct path you getting above, and HADOOP_INSTALL to your hadoop folder room

```
#HADOOP VARIABLES START  
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386  
export HADOOP_INSTALL=/usr/local/hadoop  
export PATH=$PATH:$HADOOP_INSTALL/bin  
export PATH=$PATH:$HADOOP_INSTALL/sbin  
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
```

```
export HADOOP_COMMON_HOME=$HADOOP_INSTALL  
export HADOOP_HDFS_HOME=$HADOOP_INSTALL  
export YARN_HOME=$HADOOP_INSTALL  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"  
#HADOOP VARIABLES END
```

Save file and close then Refresh Source using the following command

```
36.hduser@sysk81:~$ source ~/.bashrc  
37.hduser@sysk81:~$ javac -version  
javac 1.7.0_101  
38.hduser@sysk81:~$ which javac  
/usr/bin/javac  
hduser@sysk81:~$  
39.(2) hduser@sysk81:~$ readlink -f /usr/bin/javac  
/usr/lib/jvm/java-7-openjdk-amd64/bin/javac
```

(2) hadoop-env.sh:

```
40.hduser@sysk81:~$ nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh  
Update the JAVA_HOME  
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64  
Save and close the file
```

(3) /usr/local/hadoop/etc/hadoop/core-site.xml

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up.

This file can be used to override the default settings that Hadoop starts with.

Create a directory tmp in the data folder.

```
41.hduser@sysk81:~$ sudo mkdir -p /app/hadoop/tmp  
[sudo] password for hduser:  
42.hduser@sysk81:~$ sudo chown hduser:hadoop /app/hadoop/tmp  
hduser@sysk81:~$  
43.hduser@sysk81:~$ nano /usr/local/hadoop/etc/hadoop/core-site.xml
```

Add the following between the configuration Tag

```
<property>  
  <name>hadoop.tmp.dir</name>  
  <value>/app/hadoop/tmp</value>  
  <description>A base for other temporary directories.</description>  
</property>
```

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
</property>
```

Save and close the file

(4) /usr/local/hadoop/etc/hadoop/yarn-site.xml

```
44. hduser@sysk81:~$ nano /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

Add the following to the configuration Tag

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

Save and close the file

(5) /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the /usr/local/hadoop/etc/hadoop/ folder contains

/usr/local/hadoop/etc/hadoop/mapred-site.xml.template

file which has to be renamed/copied with the name **mapred-site.xml**:

Copy template

```
45. hduser@sysk81:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Modify file

```
hduser@sysk81:~$ nano /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Add the following to the configuration Tag

```
<property>
    <name>mapred.job.tracker</name>
    <value>localhost:54311</value>
    <description>The host and port that the MapReduce job tracker runs at. If "local", then
jobs are run in-process as a single map and reduce task.
</description>
</property>
```

Save and close the file

(6) /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The /usr/local/hadoop/etc/hadoop/hdfs-site.xml file needs to be configured for each

host in the cluster that is being used.

It is used to specify the directories which will be used as the **namenode** and the **datanode** on that host.

//Find procedure to set up the one node Hadoop cluster.

Create two directory for hadoop storage

46.hduser@sysk81:~\$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode**

47.hduser@sysk81:~\$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode**

48.hduser@sysk81:~\$ **sudo chown -R hduser:hadoop /usr/local/hadoop_store**

Modify the file hdfs-site.xml:

49.hduser@sysk81:~\$ **nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml**

Add the following to the configuration Tag

```
<property>
```

```
  <name>dfs.replication</name>
```

```
  <value>1</value>
```

```
  <description>Default block replication.
```

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

```
  </description>
```

```
</property>
```

```
<property>
```

```
  <name>dfs.namenode.name.dir</name>
```

```
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
```

```
</property>
```

```
<property>
```

```
  <name>dfs.datanode.data.dir</name>
```

```
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
```

```
</property>
```

Save and close the file

V. Change folder permission

Replace qursaan: with your hadoop users to be the owner of the folder

50.hduser@sysk81:~\$ **sudo chown hduser:hadoop -R /usr/local/hadoop**

51.hduser@sysk81:~\$ **sudo chown hduser:hadoop -R /usr/local/hadoop_store**

also give the folder the full permission

52.hduser@sysk81:~\$ **sudo chmod -R 777 /usr/local/hadoop**

```
53.hduser@sysk81:~$ sudo chmod -R 777 /usr/local/hadoop_store
```

VI. Format the New Hadoop Filesystem

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory under **/usr/local/hadoop_store/hdfs/namenode** folder:

Using Hadoop user (not superuser)

```
54.hduser@sysk81:~$ hadoop namenode -format
```

(or)

```
hduser@sysk81:~$ hdfs namenode -format
```

VII. Start Hadoop by running the following command .

Now it's time to start the newly installed single node cluster.

We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

Using Hadoop user (not superuser)

```
55.hduser@sysk81:~$ start-all.sh
```

(or)

```
cd /usr/local/hadoop/sbin
```

```
sudo su hduser
```

```
56.hduser@sysk81$ start-all.sh
```

Run **jps** coomand to see your all the services up and running:

```
hduser@sysk81:~$ jps
```

```
13337 ResourceManager
```

```
12869 NameNode
```

```
13461 NodeManager
```

```
13753 Jps
```

```
13191 SecondaryNameNode
```

```
13013 DataNode
```

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

Run **netstat -plten | grep java** to see list of ports running :

```
57.hduser@sysk81:~$ netstat -plten | grep java
```

(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)

| tcp | 0 | 0.0.0.0:50070 | 0.0.0.0:* | LISTEN | 1001 | 79069 | 12869/java |
|-----|---|------------------|-----------|--------|------|-------|------------|
| tcp | 0 | 0.0.0.0:50010 | 0.0.0.0:* | LISTEN | 1001 | 76599 | 13013/java |
| tcp | 0 | 0.0.0.0:50075 | 0.0.0.0:* | LISTEN | 1001 | 76605 | 13013/java |
| tcp | 0 | 0.0.0.0:50020 | 0.0.0.0:* | LISTEN | 1001 | 76608 | 13013/java |
| tcp | 0 | 0.127.0.0.1:9000 | 0.0.0.0:* | LISTEN | 1001 | 78066 | |

```

12869/java    tcp     0      0 0.0.0.0:50090      0.0.0.0:*
                                         LISTEN   1001    77637
13191/java    tcp6    0      0 :::41168       :::*
                                         LISTEN   1001    80980
13461/java
tcp6    0      0 :::8088       :::*
                                         LISTEN   1001    80482    13337/java
tcp6    0      0 :::13562      :::*
                                         LISTEN   1001    80991    13461/java
tcp6    0      0 :::8030      :::*
                                         LISTEN   1001    79343    13337/java
tcp6    0      0 :::8031      :::*
                                         LISTEN   1001    80943    13337/java
tcp6    0      0 :::8032      :::*
                                         LISTEN   1001    80954    13337/java
tcp6    0      0 :::8033      :::*
                                         LISTEN   1001    79570    13337/java
tcp6    0      0 :::8040      :::*
                                         LISTEN   1001    80987    13461/java
tcp6    0      0 :::8042      :::*
                                         LISTEN   1001    80992    13461/java

```

- If DataNode or Namenode is not starting, check if other programs are running in ports reserved for hadoop.

sudo netstat -tulpn | grep :8040

sudo netstat -tulpn | grep :8042

sudo netstat -tulpn | grep :50070

sudo netstat -tulpn | grep :50075

Stop Hadoop by running the following command :

We run **stop-all.sh** or (**stop-dfs.sh** and **stop-yarn.sh**) to stop all the daemons running on our machine:

58.hduser@sysk81:~\$ **stop-all.sh**

VIII. Hadoop Web Interfaces as Web View

Let's start the Hadoop again and see its Web UI:

59.hduser@sysk81:~\$ **start-all.sh**

ResourceManager @- <http://localhost:8088/>

NameNode @- <http://localhost:50070/>

SecondaryNameNode @- <http://localhost:50090/>

DataNode @- <http://localhost:50070/>

JobTracker Web Interface (MapReduce layer) <http://localhost:50030/>

TaskTracker Web Interface (MapReduce layer) <http://localhost:50060/>.

Sample Input and Output :

```
hduser@sysk42:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

```
hduser@sysk42: ~
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

16/09/23 15:59:19 INFO namenode.NameNode: STARTUP_MSG:
*****STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = sysk42/127.0.1.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 2.6.0
STARTUP_MSG: classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/
hadoop/common/lib/api-util-1.0.0-M20.jar:/usr/local/hadoop/share/hadoop/common/l
ib/commons-net-3.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beauti
ls-core-1.8.0.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-collections-
3.2.1.jar:/usr/local/hadoop/share/hadoop/common/lib/apacheds-i18n-2.0.0-M15.jar:
/usr/local/hadoop/share/hadoop/common/lib/jasper-compiler-5.5.23.jar:/usr/local/
hadoop/share/hadoop/common/lib/jetty-6.1.26.jar:/usr/local/hadoop/share/hadoop/commo
n/lib/zookeeper-3.4.6.jar:/usr/local/hadoop/share/hadoop/common/lib/jsp-api-2.1.jar:/u
srlocal/hadoop/share/hadoop/common/lib/curator-client-2.6.0.jar:/usr/local/hadoop/
share/hadoop/common/lib/curator-framework-2.6.0.jar:/usr/local/hadoop/share/hadoo
p/common/lib/snappy-java-1.0.4.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-
codec-1.4.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-core-asl-1.9.13.ja
r:/usr/local/hadoop/share/hadoop/common/lib/gson-2.2.4.jar:/usr/local/
hadoop/share/hadoop/common/lib/commons-math3-3.1.1.jar:/usr/local/hadoop/share/hadoo
p/common/lib/httpclient-4.2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/
commons-logging-1.1.3.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-el-1
.0.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/usr/l
ocal/hadoop/share/hadoop/common/lib/commons-compress-1.4.1.jar:/usr/local/hadoop/
share/hadoop/common/lib/log4j-1.2.17.jar:/usr/local/hadoop/share/hadoop/common/
lib/jsch-0.1.42.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-xc-1.9.13.
jar:/usr/local/hadoop/share/hadoop/common/lib/jersey-server-1.9.jar:/usr/local/h
adoop/share/hadoop/common/lib/jetty-6.1.26.jar:/usr/local/lib/jetty-6.1.26.jar:/u
srlocal/hadoop/share/hadoop/common/lib/xmlenc-0.52.jar:/usr/local/hadoop/share/hadoo
p/common/lib/jackson-napper-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/
lib/hadoop-auth-2.6.0.jar:/usr/local/hadoop/share/hadoop/common/lib/avro-1.7.4.ja
r:/usr/local/hadoop/share/hadoop/common/lib/asm-3.2.jar:/usr/local/hadoop/share/hadoo
p/common/lib/jasper-runtime-5.5.23.jar:/usr/local/hadoop/share/hadoop/common/lib/
servlet-api-2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-1.7.0.ja
r:/usr/local/hadoop/share/hadoop/common/lib/htrace-core-3.0.4.jar:/usr/local/hadoop/
share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/usr/local/hadoop/share/hadoop/common/lib/apacheds-kerbe
```

```
hduser@sysk38$ jps
```

```
hduser@sysk42:~$ netstat -plten | grep java
```

```
hduser@sysk42: ~
starting yarn daemons
resourceManager running as process 4486. Stop it first.
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-n
odemanager-sysk42.out
hduser@sysk42:~$ jps
5683 NodeManager
5797 Jps
5125 NameNode
4486 ResourceManager
5270 DataNode
5449 SecondaryNameNode
hduser@sysk42:~$ netstat -plten | grep java
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:50070          0.0.0.0:*      LISTEN
1001    74611  5125/java
tcp        0      0 0.0.0.0:50090          0.0.0.0:*      LISTEN
1001    78202  5270/java
tcp        0      0 0.0.0.0:50075          0.0.0.0:*      LISTEN
1001    78209  5270/java
tcp        0      0 0.0.0.0:50020          0.0.0.0:*      LISTEN
1001    78212  5270/java
tcp        0      0 127.0.0.1:54310        0.0.0.0:*      LISTEN
1001    74618  5125/java
tcp        0      0 0.0.0.0:50090          0.0.0.0:*      LISTEN
1001    78936  5449/java
tcp6       0      0 :::8088             ::::*       LISTEN
1001    69415  4486/java
tcp6       0      0 :::13562            ::::*       LISTEN
1001    78747  5683/java
tcp6       0      0 :::8030             ::::*       LISTEN
1001    69406  4486/java
tcp6       0      0 :::8031             ::::*       LISTEN
1001    69399  4486/java
tcp6       0      0 :::8032             ::::*       LISTEN
1001    69411  4486/java
tcp6       0      0 :::8033             ::::*       LISTEN
1001    69418  4486/java
tcp6       0      0 :::8040             ::::*       LISTEN
1001    78743  5683/java
tcp6       0      0 :::8042             ::::*       LISTEN
```

Name Node :

Namenode information - Mozilla Firefox

MN Running Hadoop On ... x Namenode information x All Applications x SecondaryNamenode i... x +

localhost:50070/dfshealth.html#tab-overview

Search

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Overview 'localhost:54310' (active)

| | |
|----------------|---|
| Started: | Fri Sep 23 16:01:34 IST 2016 |
| Version: | 2.6.0, re3496499ecb8d220fba99dc5ed4c99c8f9e33bb1 |
| Compiled: | 2014-11-13T21:10Z by jenkins from (detached from e349649) |
| Cluster ID: | CID-9f7c1e0d-5012-45f1-805e-06b689bac176 |
| Block Pool ID: | BP-1588801280-127.0.1.1-1474626562045 |

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 72.31 MB of 105.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 30.52 MB of 31.94 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity: 214.25 GB

Namenode information - Mozilla Firefox

MN Running Hadoop On ... x Namenode information x All Applications x SecondaryNamenode i... x +

localhost:50070/dfshealth.html#tab-overview

Decommissioning Nodes

| | |
|-----------------------------------|----------------------|
| Number of Under-Replicated Blocks | 0 |
| Number of Blocks Pending Deletion | 0 |
| Block Deletion Start Time | 23/9/2016 4:01:34 pm |

NameNode Journal Status

Current transaction ID: 3

| Journal Manager | State |
|--|--|
| FileJournalManager(root=/usr/local/hadoop_store/hdfs/namenode) | EditLogFileOutputStream(/usr/local/hadoop_store/hdfs/namenode/current/edits_inprogress_000000000000000003) |

NameNode Storage

| Storage Directory | Type | State |
|---------------------------------------|-----------------|--------|
| /usr/local/hadoop_store/hdfs/namenode | IMAGE_AND_EDITS | Active |

Hadoop, 2014.

Legacy UI

Data Node :

The screenshot shows the 'Datanode Information' page of the Hadoop web interface. The browser title bar reads 'Namenode information - Mozilla Firefox'. The main content area is titled 'Datanode Information' and displays two sections: 'In operation' and 'Decommissioning'. The 'In operation' section contains a table with one row for 'sysk42 (127.0.0.1:50010)'. The 'Decommissioning' section is currently empty. A sidebar on the left lists various Hadoop components: Namenode, SecondaryNamenode, Jobtracker, Tasktracker, Datanode, Mapred, and YARN.

| Node | Last contact | Admin State | Capacity | Used | Non DFS Used | Remaining | Blocks | Block pool used | Failed Volumes | Version |
|--------------------------|--------------|-------------|-----------|-------|--------------|-----------|--------|-----------------|----------------|---------|
| sysk42 (127.0.0.1:50010) | 0 | In Service | 214.25 GB | 24 KB | 21.45 GB | 192.8 GB | 0 | 24 KB (0%) | 0 | 2.6.0 |

Resource Manager :

The screenshot shows the 'All Applications' page of the Hadoop Resource Manager web interface. The browser title bar reads 'All Applications - Mozilla Firefox'. The main content area is titled 'All Applications' and features a large 'hadoop' logo. On the left, there is a sidebar with a tree view of cluster metrics and a table of application statistics. The table has 16 columns: Apps Submitted, Apps Pending, Apps Running, Apps Completed, Containers Running, Memory Used, Memory Total, Memory Reserved, VCores Used, VCores Total, VCores Reserved, Active Nodes, Decommissioned Nodes, and New. Below the table is a search bar and a message stating 'No data available in table'. A note at the bottom says 'Showing 0 to 0 of 0 entries'. A sidebar on the left lists components: Cluster Metrics, About, Nodes, Applications, Scheduler, and Tools.

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | New |
|----------------|--------------|--------------|----------------|--------------------|-------------|--------------|-----------------|-------------|--------------|-----------------|--------------|----------------------|-----|
| 0 | 0 | 0 | 0 | 0 | 0 B | 8 GB | 0 B | 0 | 8 | 0 | 1 | 0 | 0 |

Secondary Name node :

The screenshot shows a Mozilla Firefox browser window with the title "SecondaryNamenode Information - Mozilla Firefox". The address bar displays "localhost:50090/status.html". The main content area is titled "Overview" and contains a table with the following data:

| | |
|-------------------------|---|
| Version | 2.6.0 |
| Compiled | 2014-11-13T21:10Z by jenkins from (detached from e349649) |
| NameNode Address | localhost:54310 |
| Started | 23/9/2016 4:01:41 pm |
| Last Checkpoint | 1/1/1970 1:25:11 pm |
| Checkpoint Period | 3600 seconds |
| Checkpoint Transactions | 1000000 |

Below the table, there are sections for "Checkpoint Image URI" and "Checkpoint Editlog URI", each containing a single item:

- file:///app/hadoop/tmp/dfs/namesecondary

The screenshot shows a Mozilla Firefox browser window with the title "Directory: /logs/ - Mozilla Firefox". The address bar displays "localhost:50070/logs/". The main content area lists files in the "/logs/" directory:

| SecurityAuth-hduser.audit | 0 bytes | 23 Sep, 2016 4:01:32 PM |
|--|-------------|-------------------------|
| hadoop-hduser-datanode-sysk42.log | 22505 bytes | 23 Sep, 2016 4:01:39 PM |
| hadoop-hduser-datanode-sysk42.out | 718 bytes | 23 Sep, 2016 4:01:37 PM |
| hadoop-hduser-namenode-sysk42.log | 40849 bytes | 23 Sep, 2016 4:22:20 PM |
| hadoop-hduser-namenode-sysk42.out | 718 bytes | 23 Sep, 2016 4:01:33 PM |
| hadoop-hduser-secondarynamenode-sysk42.log | 22245 bytes | 23 Sep, 2016 4:02:44 PM |
| hadoop-hduser-secondarynamenode-sysk42.out | 718 bytes | 23 Sep, 2016 4:01:42 PM |
| userlogs/ | 4096 bytes | 23 Sep, 2016 4:21:48 PM |
| yarn-hduser-nodemanager-sysk42.log | 26277 bytes | 23 Sep, 2016 4:01:50 PM |
| yarn-hduser-nodemanager-sysk42.out | 702 bytes | 23 Sep, 2016 4:01:48 PM |
| yarn-hduser-resourcemanager-sysk42.log | 35789 bytes | 23 Sep, 2016 4:01:50 PM |
| yarn-hduser-resourcemanager-sysk42.out | 2078 bytes | 23 Sep, 2016 4:16:01 PM |

Result:

Procedure to set up the one node Hadoop cluster using Hadoop Apache open source framework with simple programming models was successfully created , executed and verified .

WRITE A WORDCOUNT PROGRAM TO DEMONSTRATE THE USE OF MAP AND REDUCE TASKS

Aim:

To write a wordcount program to demonstrate the use of Map and Reduce tasks.

Procedure :

Creating a working directory for your data

1. Create a hdfs directory for input:
 - a. hduser@sysk81:/home/panimalar\$ sudo hadoop dfs -mkdir -p
/usr/local/hadoop/input
 2. Now paste the jar file into this folder using hadoop command.
hduser@sysk81:/home/panimalar\$ sudo hadoop dfs -copyFromLocal
/usr/local/hadoop/input/4300.txt /usr/local/hadoop/input
 3. Go to \$HADOOP_HOME/share/hadoop/mapreduce folder in terminal.
cd /usr/local/hadoop/share/hadoop/mapreduce
(OR)
 - a. cd \$HADOOP_HOME/share/hadoop/mapreduce
Here /usr/local/hadoop/output folder will be created. Each time you have to give a new name.
- NOTE:** you cannot simply list the folders input and output using ls in terminal.
4. To view the output, use this:
hduser@sysk81:/home/panimalar\$ **hadoop dfs -cat**
/usr/local/hadoop/output/part-r-00000

Commands :

// Change the Directory

hduser@sysk81:~\$ **cd /usr/local/hadoop**

// Make the input directory in /usr/local/hadoop

hduser@sysk81:~\$**mkdir input**

// Change the directory into input directory

hduser@sysk81:~\$**cd /usr/local/hadoop/input**

// Download the sample Bigdata input file

hduser@sysk81:~\$**wget <http://www.gutenberg.org/files/4300/4300.zip>**

// Unzip the compressed file

hduser@sysk81:~\$**unzip 4300.zip**

// Remove the Downloaded file after unzip it

```

hduser@sysk81:~$rm 4300.zip
// Copy the data file 4300.txt to the Hadoop File System (HDFS)
hduser@sysk81:~$ cp 4300.txt /usr/local/hadoop/input
// Check the contents of your directory
hduser@sysk81:~$ sudo hadoop dfs -ls
// Output :
Found 1 items
-rw-r--r-- 1 hduser hadoop 1573079 2014-12-09 03:08 4300.txt
// Copy any file from the web
hduser@sysk81:/usr/local/hadoop/input$curl
http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html > setup.html
//Copy one or more text files into the input directory
panimalar@sysk81:/usr/local/hadoop/input$ sudo hadoop fs -copyFromLocal
setup.html input
( or )
panimalar@sysk81:~$ hadoop fs -copyFromLocal setup.html input
// Change the directory into mapreduce
panimalar@sysk81:~$ cd /usr/local/hadoop/share/hadoop/mapreduce

```

Program :

```

package wc;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.util.*;
public class WordCount extends Configured implements Tool
{
    public static void main(String args[]) throws Exception

```

```

{
    //this main function will call run method defined above.
    int res = ToolRunner.run(new WordCount(), args);
    System.exit(res);
}

public int run(String[] args) throws Exception
{
    Path inputPath = new Path(args[0]);
    Path outputPath = new Path(args[1]);
    //creating a Job and Conf object and assigning a job name for identification purposes
    Configuration conf = getConf();
    Job job = new Job(conf, this.getClass().toString());
    //The hdfs input and output directory to be fetched from the command line
    FileInputFormat.setInputPaths(job, inputPath);
    FileOutputFormat.setOutputPath(job, outputPath);
    job.setJobName("WordCount");
    job.setJarByClass(WordCount.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(IntWritable.class);
    //Setting configuration object with the Data Type of output Key and Value

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

// The map-reduce organization section
    job.setMapperClass(Map.class);
    job.setCombinerClass(Reduce.class);
    job.setReducerClass(Reduce.class);
    return job.waitForCompletion(true) ? 0 : 1;
}

//The map section
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>
{
    //hadoop supported data types
    private final static IntWritable one = new IntWritable(1);
}

```

```

private Text word = new Text();
//map method that performs the tokenizer job and framing the initial key value pairs
// after all lines are converted into key-value pairs, reducer is called.
@Override
public void map(LongWritable key, Text value, Mapper.Context context) throws
IOException, InterruptedException
{
//taking one line at a time from input file and tokenizing the same
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
//iterating through all the words available in that line and forming the key value pair
    while (tokenizer.hasMoreTokens())
    {
        word.set(tokenizer.nextToken());
        context.write(word, one);
    }
}
// The reduce section
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>
{
//reduce method accepts the Key Value pairs from mappers, do the aggregation based on keys and
produce the final out put
@Override
public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException
{
    int sum = 0;
/*iterates through all the values available with a key and add them together and give the
final result as the key and sum of its values*/
    for (IntWritable value : values)
    {
        sum += value.get();
    }
    context.write(key, new IntWritable(sum));
}

```

```
}
```

```
}
```

To Run :

// To run the job on data in the input directory, and place results in the output directory.

```
panimalar@sysk81:/usr/local/hadoop/share/hadoop/mapreduce$ sudo hadoop jar hadoop-mapreduce-examples-2.6.0.jar wordcount /usr/local/hadoop/input/4300.txt /usr/local/hadoop/output
```

// See the Output :

```
panimalar@sysk81:~$ sudo hadoop dfs -cat /usr/local/hadoop/output/part-r-00000
```

Sample Input and Output :

```
13:22:30 INFO mapreduce.Job: map 100% reduce 100%
13:22:30 INFO mapreduce.Job: Job job_1391412385921_0002 completed successfully
13:22:31 INFO mapreduce.Job: Counters: 43
File System Counters
    FILE: Number of bytes read=89
    FILE: Number of bytes written=160142
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=171
    HDFS: Number of bytes written=59
    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)=5657
    Total time spent by all reduces in occupied slots (ms)=6128
Map-Reduce Framework
    Map input records=2
    Map output records=7
    Map output bytes=82
    Map output materialized bytes=89
    Input split bytes=116
    Combine input records=7
    Combine output records=6
    Reduce input groups=6
    Reduce shuffle bytes=89
    Reduce input records=6
    Reduce output records=6
    Spilled Records=12
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=145
    CPU time spent (ms)=1418
    Physical memory (bytes) snapshot=368246784
    Virtual memory (bytes) snapshot=513716224
    Total committed heap usage (bytes)=307757056
Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
```

```
$ hdfs dfs -cat output/*
```

Example 1
Hadoop 2
Install 1
Mapreduce 1
Run 1
Wordcount 1

The screenshot shows the Hadoop Web UI interface. At the top, there's a navigation bar with links for 'All Applications', 'Cluster Metrics', 'Nodes', and 'Jobs'. Below the navigation is a search bar and a logo for 'hadoop'. The main area is titled 'All Applications' and displays a table of running applications. The table has columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking URL. Two rows are visible: one for 'application_1391412385921_0002' and another for 'application_1391412385921_0001'. Both rows show a 'FINISHED' state and 'SUCCEEDED' final status. The tracking URLs for both are highlighted with a red box.

| ID | User | Name | Application Type | Queue | Start Time | Finish Time | State | Final Status | Progress | Tracking URL |
|--------------------------------|---------|------------|------------------|---------|-------------------------------|-------------------------------|----------|--------------|----------|--------------|
| application_1391412385921_0002 | ABHUITG | word count | MAPREDUCE | default | Mon, 03 Feb 2014 07:52:04 GMT | Mon, 03 Feb 2014 07:52:29 GMT | FINISHED | SUCCEEDED | | History |
| application_1391412385921_0001 | ABHUITG | word count | MAPREDUCE | default | Mon, 03 Feb 2014 07:38:43 GMT | Mon, 03 Feb 2014 07:39:15 GMT | FINISHED | SUCCEEDED | | History |

Result:

Hence the wordcount program to demonstrate the use of map and reduce tasks is successfully executed.

SIMULATE A CLOUD SCENARIO USING CLOUDSIM AND RUN A SCHEDULING ALGORITHM THAT IS NOT PRESENT IN CLOUDSIM

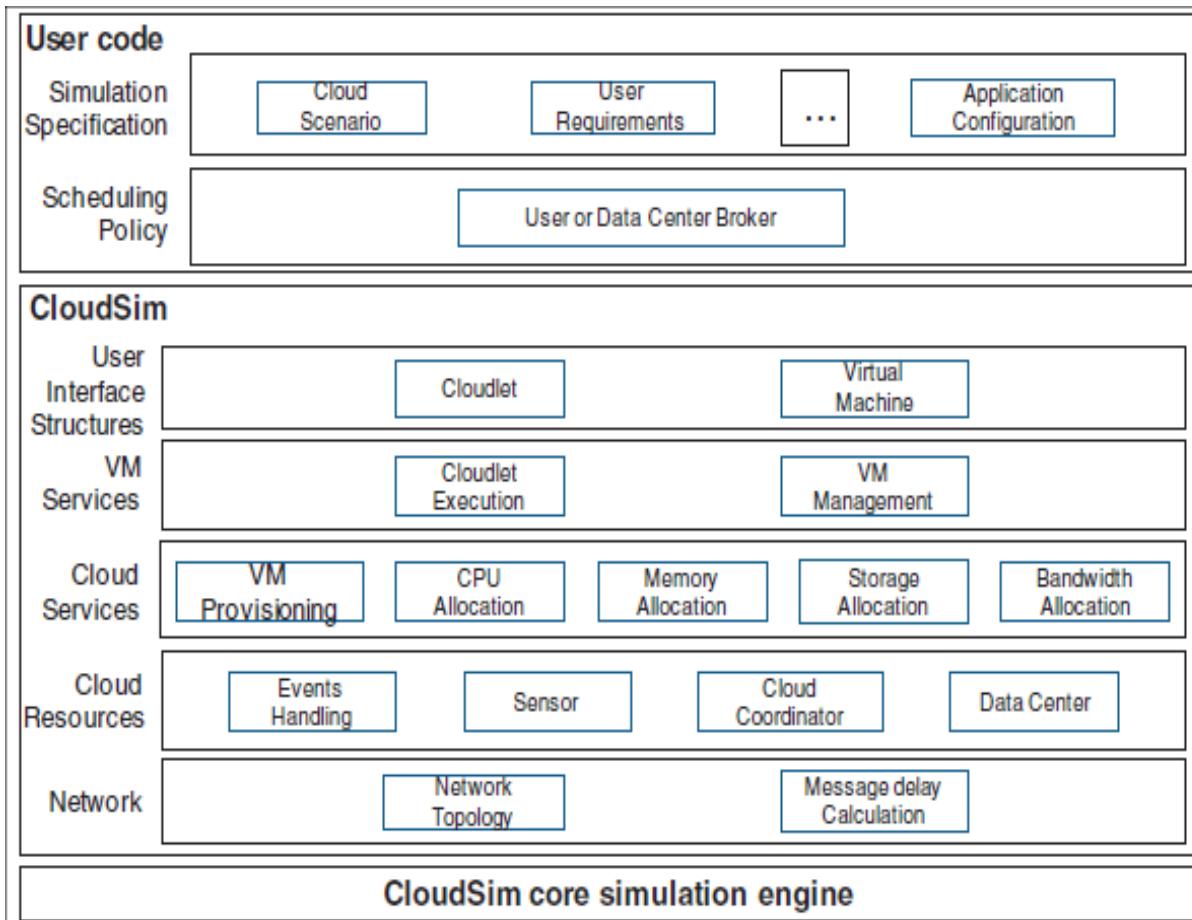
Aim:

To Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim .

Procedure :

CloudSim is a programming language based simulator and even though it does not support a graphical user interface for simulation.

CloudSim is a library for the simulation of cloud scenarios . CloudSim support for modeling and simulation of large scale Cloud computing environments, including data centers, on a single physical computing node.



CloudSim offers architecture inside four uncomplicated entities. these types of entities offer consumer to set-up the basic cloud computing environment as well as measure your effectiveness involving fill up Balancing algorithms.. Datacenters entity features the responsibility of providing Infrastructure level solutions for the Cloud Users. They act as a home to help a lot of Host Entities or maybe a lot of instances hosts' entities aggregate to help application form the solitary Datacenter entity. Hosts with Cloud are usually Physical Servers

Scheduling in Cloud :

- As **cloud computing** is the **virtualized operating environment**, and the **virtual machines** are the **primary computing component** which is **responsible for the execution of the workloads(tasks)**. The **virtual machine(s)** are powered by a **physical server host machine (i.e.) hardware**.
- Depending on the requirement of the Virtual Machine(VM) there could be '**one to one**' or '**many to one**' **mapping between the VM and host machine**. That means in cloud computing the scheduling is done at both the mapping levels that are:
 - **Virtual Machine to Host Machines**
 - **Tasks to Virtual Machines**

Steps :

1. Setting the number of users for a current simulation
2. Initializing the simulation by instantiating the common variables (current time, trace flag, number of users)
3. Creating CIS instance
4. Creating data center instance and then registering it with CIS
5. Creating physical machines (hosts) with their characteristics
6. Creating data center broker instance
7. Creating VMs with their characteristics
8. Submitting VMs to data center broker
9. Creating cloudlets and specifying their characteristics
10. Submitting cloudlets to data center broker.
11. Sending a call to start the simulation once there is an event to be executed
12. Sending a call to stop the simulation once there is no event to be executed

13. Printing the results of the simulation

Design of Cloudsim :

There are several classes need to modify if want to implement own algorithms

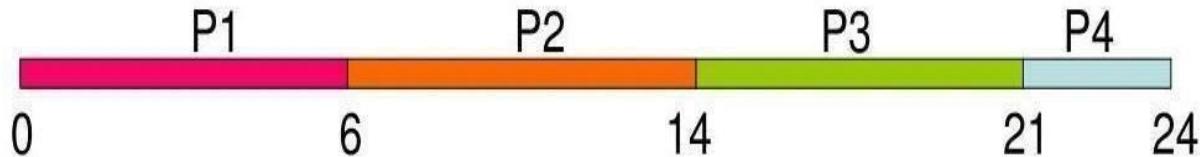
- DatacenterBroker - Modifying the way VM provisioning requests are submitted to data centers and the way cloudlets are submitted and assigned to VMs.
- VmAllocationPolicy - Need to extend this abstract class to implement your own algorithms for deciding which host a new VM should be placed on
- VmScheduler - Implementing algorithms for resource allocation to VMs within a single host.
- CloudletScheduler - Implementing algorithms for scheduling cloudlets within a single VM

Shortest Job First Scheduling :

In SJF, Process from the ready queue that has shortest finish time will execute first. If two process are having same burst time and arrival time, then FCFS procedure is used to break the tie.

FCFS Example

| Process | Duration | Priority | Arrival Time |
|---------|----------|----------|--------------|
| P1 | 6 | 4 | 0 |
| P2 | 8 | 1 | 0 |
| P3 | 7 | 3 | 0 |
| P4 | 3 | 2 | 0 |



Program :

Shortest Job First (SJF) Task scheduling algorithm implementation in cloudsim

FCFS.Java

```
package org.cloudbus.cloudsim; import java.text.DecimalFormat; import java.util.Calendar; import  
java.util.List;  
import org.cloudbus.cloudsim.Cloudlet; import org.cloudbus.cloudsim.Datacenter; import  
org.cloudbus.cloudsim.Log; import org.cloudbus.cloudsim.Vm;  
import org.cloudbus.cloudsim.core.CloudSim;  
/**  
 * FCFS Task scheduling  
 */  
public class FCFS {  
    /** The cloudlet list. */  
    private static List<Cloudlet> cloudletList;  
    /** The vmlist. */  
    private static List<Vm> vmlist;  
    private static int reqTasks = 10; private static int reqVms = 4;  
    /**  
     * Creates main() to run this example  
     */  
    public static void main(String[] args) { Log.printLine("Starting FCFS...");  
        try {  
            // First step: Initialize the CloudSim package. It should be called  
            // before creating any entities.  
            int num_user = 1; // number of cloud users  
            Calendar calendar = Calendar.getInstance();  
            boolean trace_flag = false; // mean trace events  
            // Initialize the CloudSim library  
            CloudSim.init(num_user, calendar, trace_flag);  
            // Second step: Create Datacenters  
            //Datacenters are the resource providers in CloudSim. We need at least one of them to run a CloudSim  
            simulation  
            @SuppressWarnings("unused")  
            Datacenter datacenter0 = createDatacenter("Datacenter_0");  
            //Third step: Create Broker  
            FcfsBroker broker = createBroker();  
            int brokerId = broker.getId();  
            //Fourth step: Create one virtual machine  
            vmlist = new VmsCreator().createRequiredVms(reqVms, brokerId);  
            //submit vm list to the broker  
            broker.submitVmList(vmlist);  
            //Fifth step: Create two Cloudlets  
            cloudletList = new CloudletCreator3().createUserCloudlet(reqTasks, brokerId);  
            //submit cloudlet list to the broker  
            broker.submitCloudletList(cloudletList);  
            //call the scheduling function via the broker  
            broker.scheduleTasksToVms(); // Sixth step: Starts the  
            simulation  
            CloudSim.startSimulation();  
            // Final step: Print results when simulation is over  
            List<Cloudlet> newList =  
                broker.getCloudletReceivedList();  
            CloudSim.stopSimulation();  
            printCloudletList(newList);  
            Log.printLine("FCFS finished!");  
        }  
        catch (Exception e) { e.printStackTrace();  
            Log.printLine("The simulation has been terminated due to an unexpected error");  
        }  
    }
```

```
}

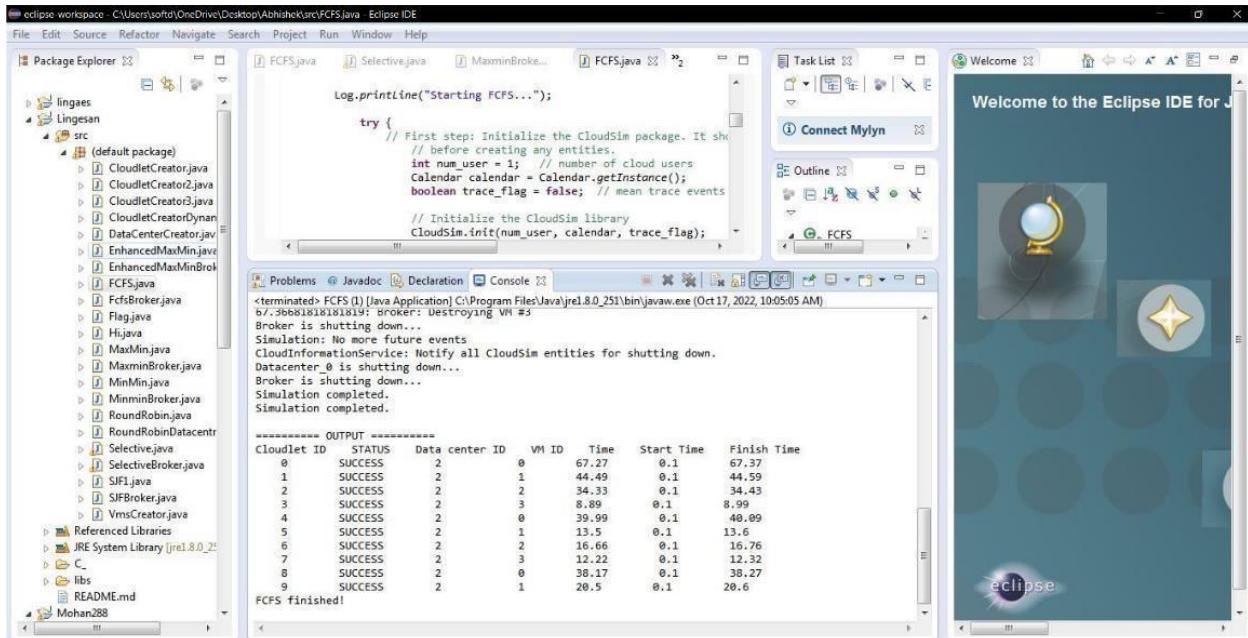
private static Datacenter createDatacenter(String name){
    Datacenter datacenter=new DataCenterCreator().createUserDatacenter(name, reqVms); return
    datacenter;
}

//We strongly encourage users to develop their own broker policies, to submit vms and cloudlets
according
//to the specific rules of the simulated scenario private static FcfsBroker createBroker(){
FcfsBroker broker = null; try {
    broker = new FcfsBroker("Broker");
} catch (Exception e) { e.printStackTrace(); return null;
}
return broker; }

/***
 * Prints the Cloudlet objects
 * @param list list of Cloudlets

*/
private static void printCloudletList(List<Cloudlet> list) { int size = list.size();
Cloudlet cloudlet;
String indent = " "; Log.println();
Log.println("===== OUTPUT =====");
Log.println("Cloudlet ID" + indent + "STATUS" + indent +
"Data center ID" + indent + "VM ID" + indent + "Time" + indent + "Start Time" + indent + "Finish
Time");
DecimalFormat dft = new DecimalFormat("##.##"); for (int i = 0; i < size; i++) {
    cloudlet = list.get(i);
    Log.print(indent + cloudlet.getCloudletId() + indent + indent);
    if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){ Log.print("SUCCESS");
        Log.println( indent + indent + cloudlet.getResourceId() + indent + indent + indent +
        cloudlet.getVmId() +
        indent + indent + dft.format(cloudlet.getActualCPUTime()) + indent + indent +
        dft.format(cloudlet.getExecStartTime())+
        indent + indent + dft.format(cloudlet.getFinishTime())));
    }
}
}
```

Sample Output :



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like Lingean, Lingean, and src containing various Java files such as CloudletCreator.java, DataCenterCreator.java, EnhancedMaxMin.java, FCFS.java, FcfsBroker.java, Flag.java, Hijava, MaxMin.java, MaxminBroker.java, MinMin.java, MinminBroker.java, RoundRobin.java, RoundRobinDatacentr.java, Selective.java, SelectiveBroker.java, SIF.java, SIFBroker.java, and VmsCreator.java.
- Code Editor:** Displays the FCFS.java code, which initializes the CloudSim package and runs a simulation.
- Console:** Shows the simulation output, including log messages and a summary table of cloudlet execution results.
- Welcome:** A panel on the right displays the Eclipse logo and some decorative icons.

```
Log.println("Starting FCFS...");

try {
    // First step: Initialize the CloudSim package. It should be done before creating any entities.
    int num_user = 1; // number of cloud users
    Calendar calendar = Calendar.getInstance();
    boolean trace_flag = false; // mean trace events

    // Initialize the CloudSim library
    CloudSim.init(num_user, calendar, trace_flag);
}

<terminated> FCFS (I) [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (Oct 17, 2022, 10:05:05 AM)
b73bb81818181819: Broker: Destroying VM #3
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

=====
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time
0 SUCCESS 2 0 67.27 0.1 67.37
1 SUCCESS 2 1 44.49 0.1 44.59
2 SUCCESS 2 2 34.33 0.1 34.43
3 SUCCESS 2 3 8.89 0.1 8.99
4 SUCCESS 2 0 39.99 0.1 40.09
5 SUCCESS 2 1 13.5 0.1 13.6
6 SUCCESS 2 2 16.66 0.1 16.76
7 SUCCESS 2 3 12.22 0.1 12.32
8 SUCCESS 2 0 38.17 0.1 38.27
9 SUCCESS 2 1 20.5 0.1 20.6

FCFS finished!
```

| Cloudlet ID | STATUS | Data center ID | VM ID | Time | Start Time | Finish Time |
|-------------|---------|----------------|-------|-------|------------|-------------|
| 0 | SUCCESS | 2 | 0 | 67.27 | 0.1 | 67.37 |
| 1 | SUCCESS | 2 | 1 | 44.49 | 0.1 | 44.59 |
| 2 | SUCCESS | 2 | 2 | 34.33 | 0.1 | 34.43 |
| 3 | SUCCESS | 2 | 3 | 8.89 | 0.1 | 8.99 |
| 4 | SUCCESS | 2 | 0 | 39.99 | 0.1 | 40.09 |
| 5 | SUCCESS | 2 | 1 | 13.5 | 0.1 | 13.6 |
| 6 | SUCCESS | 2 | 2 | 16.66 | 0.1 | 16.76 |
| 7 | SUCCESS | 2 | 3 | 12.22 | 0.1 | 12.32 |
| 8 | SUCCESS | 2 | 0 | 38.17 | 0.1 | 38.27 |
| 9 | SUCCESS | 2 | 1 | 20.5 | 0.1 | 20.6 |

Fig. Configuration details of Cloudsim simulator

Result :

The Simulation of a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim was successfully learned and verified.