KAMARAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution - AFFILIATED TO ANNA UNIVERSITY, CHENNAI)

S.P.G.Chidambara Nadar - C.Nagammal Campus

S.P.G.C. Nagar, K.Vellakulam – 625 701 (Near VIRUDHUNAGAR).

DEPARTMENT

OF

INFORMATION TECHNOLOGY

B.Tech IT

LABORATORY MANUAL CUM RECORD

(AS PER OUTCOME BASED EDUCATION - OBE)

Cs1781 - Cloud Computing Laboratory

KAMARAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution - AFFILIATED TO ANNA UNIVERSITY, CHENNAI)
S.P.G.Chidambara Nadar - C.Nagammal Campus
S.P.G.C. Nagar, K.Vellakulam – 625 701 (Near VIRUDHUNAGAR).

DEPARTMENT : _____

NAME : _____

REGISTER NUMBER : _____

ROLL NO : _____

BRANCH : _____

YEAR & SECTION : _____

SEMESTER : _____

Bona-fide Record of work done in the _____ of Kamaraj College of Engineering and Technology, Near Virudhunagar, during the Academic Year 2023 - 2024

COURSE INCHARGE **HEAD OF THE DEPARTMENT**

Submitted for the Practical Examination held on _____ at Kamaraj College of Engineering and Technology, Near Virudhunagar.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

CONTENT

INDEX						
Ex.No	**Date**	**Experiment Name**	**PageNo**	**Marks (100)**	**CO Attained? (Yes/No)**	**Sign**
1a		Install Virtualbox with different flavours of OS				
1b		Install a C compiler in the virtual machine				
2		Create instances and volume attachment in AWS				
3		Setup a Private Cloud Using Open Stack or Eucalyptus				
4		Install Open Stack Object Storage -Swift in Ubuntu				
5		Implement OpenStack Nova – Compute and Image services –Glance.				
6		Setup Openstack Using DevStack				
7		Install a Single node hadoop cluster				
8		Implement Map Reduce concept for word count program				
9		Implement VM Scheduling Using CloudSim				
10		Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories using Git Bash and Git Hub.				

Laboratory Specific Instructions to Students

DOs:

1. Remove your footwear, outside the lab in its place.
2. Get permission before entering the lab.
3. Wear lab coat and ID Card while in Lab.
4. Record the use of system in the login register.
5. Report all problems to faculty in-charge.
6. Login in to the system with your credential only.
7. Remain seated in your allotted place and work quietly.
8. In case of any doubt ask the available teaching faculty.
9. Save all your work in the server with your login credential.
10. Keep your lab neat and clean.
11. Logoff and Properly shutdown the system after usage.
12. Switch off power supply (after proper shutdown) of the system and arrange your chairs before leaving the laboratory.
13. Use internet facility, with the permission of faculty, only for educational purpose.

Dont's:

1. Possession of eatables, drinks and own baggage inside the lab is prohibited.
2. Sharing of username and password is not allowed.
3. Don't Change the Computer settings.
4. Never attempt to fix a system problem on your own.
5. Moving any equipment from its original position is prohibited.
6. Usage of pen drive/mobile phone storage/external disk in laboratory is restricted.
7. Throwing garbage in the laboratory is not allowed.
8. Loitering in the laboratory is restricted.
9. Making noise in the laboratory is strictly prohibited.
10. Don't use internet facility in the lab without the permission of faculty.

Section: 1 Course Introductions

Introduction of Lab Course

Course Code (As per Curriculum)	:	**CS1781**
Course Code (As per NBA)	:	**20CSC409**
Course Name	:	**Cloud Computing Laboratory**

Importance of Course (150 Words write up)

a. **Scalability:** Cloud computing allows you to use as many or as few resources as you need. Therefore, depending on your business needs or projected traffic to your business you can choose to increase or decrease your investment in IT infrastructure

b. **Saving Costs:** Cloud computing helps businesses to reduce costs in various ways. Companies only pay for the resources they use, making this process a more economical option than having to buy and manage their own resources. Cloud computing also results in considerable savings in Capital Expenditure and Operating Expenditure because companies do not have to invest in expensive hardware, storage devices, software, etc.

c. **Disaster Recovery:** With all data stored in the 'cloud' backup and recovery of data and applications is quicker and more reliable. This applies to all sizes of organizations and volumes of data. 20% of cloud users claim disaster recovery in four hours or less as opposed to only 9% of non-cloud users.

d. **Security:** It is the duty and responsibility of the cloud service providers to carefully monitor security. Compare this against an in-house I.T. department, for

Syllabus – List of Experiments

Course Code (As per Curriculum)	:	**CS1781**
Course Code (As per NBA)	:	**20CSC409**
Course Name	:	**Cloud Computing Laboratory**

Aim :

- To learn and develop applications using gcc and make
- To learn and use version control systems
- To develop web applications in cloud

Objectives :

- To learn the design and development process involved in creating a cloud-based application
- To learn to implement and use parallel programming using Hadoop

List of Experiments :

1a Install Virtualbox with different flavours of OS
1b Install a C compiler in the virtual machine
2 Create instances and volume attachment in AWS
3 Setup a Private Cloud Using Open Stack or Eucalyptus
4 Install Open Stack Object Storage -Swift in Ubuntu.
5 Implement OpenStack Nova – Compute and Image services –Glance
6 Setup Openstack Using DevStack
7 Install a Single node hadoop cluster
8 Implement Map Reduce concept for word count program
9 Implement VM Scheduling Using CloudSim
10 Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories using Git Bash and Git Hub.

Section: 2 Outcome Based Education (OBE)

VISION & MISSION & QUALITY POLICY OF THE COLLEGE

Vision	:	To Make this Institution the unique of its kind in the field of **Research and Development activities** in this part of world
Mission	:	To impart highly innovative and technical knowledge to the urban and unreachable rural student folks through **"Total Quality Education".**
Quality Policy	:	Committed to impart Quality Technical Education imbibed with **proficiency, human values** and **continual improvement**.

VISION & MISSION OF THE DEPARTMENT

Vision	:	To make the Department of Information Technology, the unique of its kind in the field of **Research and Development activities** in this part of world
Mission	:	To impart highly innovative and technical knowledge in the field of Information Technology to the urban and unreachable rural student folks through **"Total Quality Education".**

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

PEO -1	:	Technical Knowledge : Graduates will be able to identify, analyze and create solutions for real life industrial and societal needs by applying the principles and practices of Information Technology.
PEO -2	:	Teamwork& Ethics : Graduates will be able to collaborate effectively and ethically in a multi-disciplinary team as a member & / as a leader.
PEO -3	:	Lifelong Learning : Graduates will be able to adopt the contemporary technologies in the field of Information Technology to provide solutions for challenging environments.

OBE- Outcome Based Education - Introduction

Aim:

i. Outcomes are more **like signboards and roadmaps** to **help the learners** reach where **they're supposed to reach**, and **contribute to progress.**

ii. **Outcome measurements** provide the basis for **continuous improvement in the quality of learning**

Terminology	Description
Accreditation	i. With the introduction of the **washington accords**, our **higher education demands** better, intelligent workflows for the sake of **improving quality** ii. The accreditation and regulatory organisations including NBA, NAAC, AICTE etc. has clearly made their intentions clear about **introducing a skill-based, competency nurturing learning experience** iii. As of the **NBA**, their accreditation methods and **assessment** parameters **are based on OBE**
OBE- Outcome Based Education	i. Deciding outcomes for academic achievements and it's attainment for assessment and formulation is based **on a learning theory called** Outcome Based Education**(OBE)** ii. OBE is **an educational theory** that bases each part of an educational system **around goals (outcomes**). iii. **Student role**: By the **end of the educational experience**, each student **should have achieved the goal**. There is no single specified style of teaching or assessment in OBE; instead, classes, opportunities, and assessments should all **help students achieve the specified outcomes.** iv. **Faculty role**: The role of the **faculty adapts** into instructor, trainer, facilitator, and/or mentor **based on the outcomes targeted.**
OBE emphasizes on	i. Stating **what you want your students to be able to do** at the end of the program, ii. Assessing the students **whether they are able to do** what they are expected to do to do what they are expected to do, iii. Orienting **teaching** and other academic processes to **facilitate students to do** what they are expected to do. Note: Outcomes are different from your course objectives.
Course Objectives	i. An objective is more **like a plan**, or **a road map** to reach/attain the outcomes. ii. A course objective describes what a faculty member will cover in a course. They are **generally less broader than desirable goals** and **more broader than** student **learning outcomes.**
Course Outcomes	i. **Outcome** — A detailed description of what a student must be able to do at the conclusion of a course.

(COs)

ii. POs are attained through program specific Core Courses, which has their own, previously set outcomes to attain. These course-specific outcomes are called Course Outcomes.

iii. **No.of CO statement:** Each course is designed to meet (about 5–6) Course Outcomes.

iv. **Structure of COs Statements:** The Course Outcomes are stated in such a way that they can be actually measured. (**SMART –"**S- Specific; M-Measurable; A-Achievable; R-Realistic; T-Time bound")

v. **Framing of COs:** COs are set by the institution, by consulting with the department heads, faculty, students and other stakeholders.

Program Outcomes (POs)

i. POs **are statements** about the knowledge, skills and attitudes (attributes) the graduate of a formal engineering program should have.

ii. POs **deal with** the general aspect of graduation for a particular program, and the **competencies and expertise a graduate will possess after completion of the program**.

iii. These are broad and cover a wider area than of COs. the NBA has set **12 Program Outcomes** or Graduate Attributes **for the sake of unity and quality assurance.**

Mapping of CO vs PO

i. Map the objective to the **outcomes** to analyse and **document their attainment**

ii. Each CO can be identified to **address a subset of POs**

iii. Based on the number of COs and sessions dedicated to them , it is possible to identify the **strength** of **mapping (1,2 or 3) to POs**

iv. Based on these strength of selected POs a CPO matrix can be established.

PO matrix can be established

Course	PO1	PO2	PO3			PO11	PO12
CO1	1	0	3	1	2	3	2

Program Educational Objectives (PEOs)

i. Program Educational Objectives (PEO) *are* **statements** that describe the career and professional accomplishments that the **program is preparing the graduates to achieve**.

ii. Measuring PEOs: PEO's are measured **4–5 years after graduation.**

iii. They are set in order to measure the effectiveness of the program, and to check **whether it has prepared the students to deal with the real world,** where they could apply and use the skills and knowledge they've learned to good use.

iv. **No.of PEOs**: Each program shall specify 2–4 program specific outcomes for the accreditation by the NBA.

CO Attainment

CO Assessment:

i. The assessments should be in alignment with the COs

ii. Question paper should be so set to assess all COs

iii. The average marks obtained in assessments against items for each CO will indicate the CO attainment

CO Attainment gaps:

iv. Instructors can set targets for each CO of his/her course

v. Attainment gaps can therefore be identified

vi. Instructor can plan to reduce the attainment gaps or enhance attainment targets enhance attainment targets

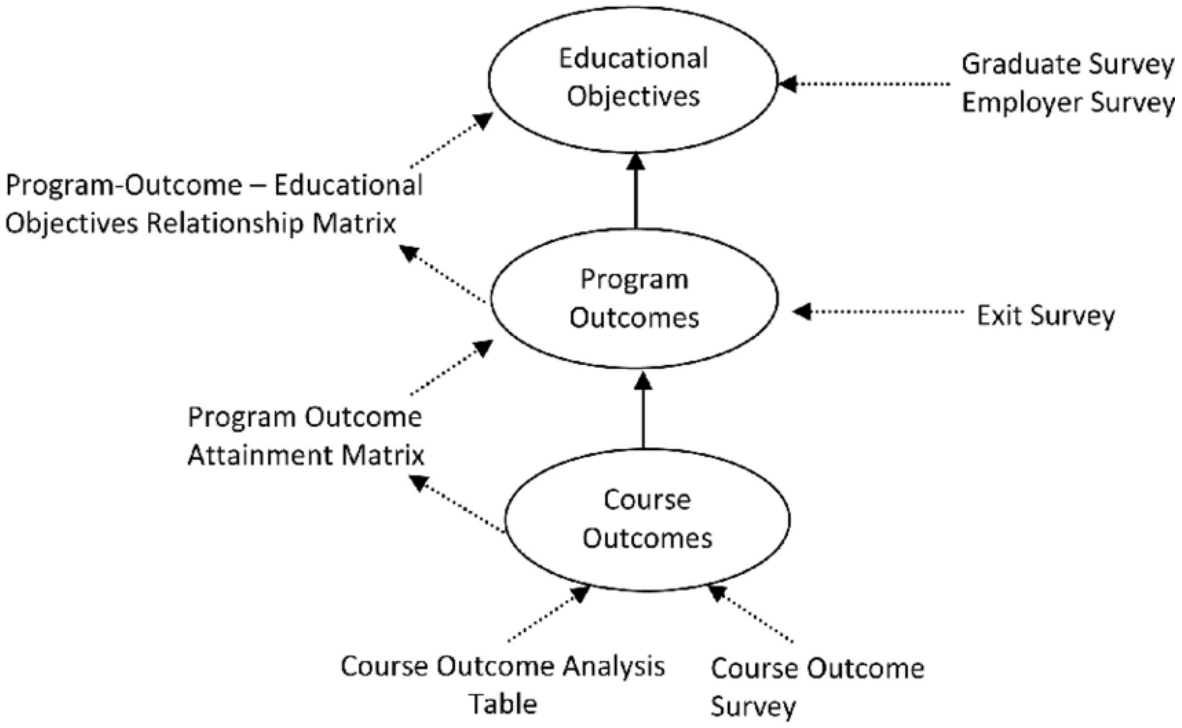
CO Attainment:

	Program Outcome (PO) (*Graduate Attributes)*		
PO/ PSO	**Engineering Graduate will be able to :**	**Learning Level**	**At end of the laboratory course, attained the Outcomes/Skill of the student- (Remark by students)**
PO 1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization for the solution of complex engineering problems.	**K3 - Apply**	
PO 2	**Problem analysis:** Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.	**K4 - Analyze**	
PO 3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.	**K5 - Evaluate**	
PO 4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	**K5 - Evaluate**	
PO 5	**Modern tool usage:** Create, select and apply appropriate techniques, resources and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.	**K6 - Create**	

PO 6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.	**K3 - Apply**	
PO 7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	**K2 Understand**	
PO 8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	**K3 - Apply**	
PO 9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	**A3 -Value**	
PO 10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	**A3 - Value**	
PO 11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	**K3 - Apply**	
PO 12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.	**A5 Characterize**	

	Program Specific Outcome (PSOs) At the end of the program, the student		
PSO 1	Demonstrate technical and interpersonal skills to design and develop IT enabled solutions to meet the real time industrial and societal needs	**K4 - Analyze**	
PSO 2	Exhibit an ability to adapt to the evolutionary changes in computing	**K5 - Evaluate**	

Flow Chart -Attainment of Program Educational Objectives (PEO's) through PO & CO



Course Outcome (COs)

Course Code (As per Curriculum)	:	CS1781
Course Code (As per NBA)	:	**20CSC409**
Course Name	:	Cloud Computing Laboratory

Course Objectives

- To develop web applications in cloud
- To learn to implement and use parallel programming using Hadoop
- To learn to implement the version control commands using Github/Gitbash

Course Outcomes

On successful completion of this course, students will be able to

CO	CO-Statements	Knowledge Level
CO1	Experiment with virtual machines from available physical resources	K3 - Apply
CO2	Experiment with virtual machines using Openstack/Eucalyptus.	K3 - Apply
CO3	Build single node Hadoop cluster	K3 - Apply
CO4	Apply task scheduling algorithms in Cloudsim	K3 - Apply
CO5	Make use of version control commands for file management	K3 - Apply

Laboratory Assessment Tool

S.No	Assessment Tool	Weightage (%)
1	Continuous Assesment	75%
2	Model Exam	25%

Target Value = 80 % of Marks

CO –PO Mapping														
	Program outcomes												***Program Specific outcomes***	
CO	***PO1*** **(K3)**	***PO2*** **(K4)**	***PO3*** **(K5)**	***PO4*** **(K5)**	***PO5*** **(K6)**	***PO6*** **(K3)**	***PO7*** **(K2)**	***PO8*** **(K3)**	***PO9*** **(A3)**	***PO10*** **(A3)**	***PO11*** **(K3)**	***PO12*** **(K5)**	***PSO1*** **(K3)**	***PSO2*** **(K5)**
CO1	L	L	M	L	H	L	-	-	L	L	-	L	M	M
CO2	M	L	M	M	H	L	-	-	L	L	-	M	M	M
CO3	H	M	M	M	H	L	-	-	L	L	-	M	M	M
CO4	M	L	M	M	H	L	-	-	L	L	-	M	M	M
CO5	M	M	M	M	H	L	-	-	L	L	-	M	M	M

CO –PO Matrix														
	Program outcomes												***Program Specific outcomes***	
CO	***PO1*** **(K3)**	***PO2*** **(K4)**	***PO3*** **(K5)**	***PO4*** **(K5)**	***PO5*** **(K6)**	***PO6*** **(K3)**	***PO7*** **(K2)**	***PO8*** **(K3)**	***PO9*** **(A3)**	***PO10*** **(A3)**	***PO11*** **(K3)**	***PO12*** **(K5)**	***PSO1*** **(K3)**	***PSO2*** **(K5)**
CO1	1	1	2	1	3	1	-	-	1	1	-	1	2	2
CO2	2	1	2	2	3	1	-	-	1	1	-	2	2	2
CO3	3	2	2	2	3	1	-	-	1	1	-	2	2	2
CO4	2	1	2	2	3	1	-	-	1	1	-	2	2	2
CO5	2	2	2	2	3	1	-	-	1	1	-	2	2	2
PO Expected (Round off Average)	**1.8**	**1.4**	**2**	**1.8**	**3**	**1**	**-**	**-**	**1**	**1**	**-**	**1.8**	**2**	**2**

Note: **Target Value** = 80 %

If CO Attainment,

YES = Laboratory Mark (%) >= Target Value (%); **NO**= Laboratory Mark (%) < Target Value (%)

LIST OF EXPERIMENTS & ITS CO					
S.No	**Experiment Name**	**Course Outcome**	**Blooms Taxonomy level**	**Marks Obtained Out of 100**	**CO Attained? (Y/N)**
1a	Install Virtualbox with different flavours of OS	CO1	K3		
1b	Install a C compiler in the virtual machine	Co1	K3		
2	Create instances and volume attachment in AWS	CO1	K3		
3	Setup a Private Cloud Using Open Stack or Eucalyptus	CO2	K3		
4	Install Open Stack Object Storage -Swift in Ubuntu.	CO2	K3		
5	Implement OpenStack Nova – Compute and Image services – Glance.	CO2	K3		
6	Setup Openstack Using DevStack.	CO2	K3		
7	Install a Single node hadoop cluster	CO3	K3		
8	Implement Map Reduce concept for word count program	CO3	K3		
9	Implement VM Scheduling Using CloudSim	CO4	K3		
10	Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories using Git Bash and Git Hub.	CO5	K3		

CO ATTAINMENT - Internal Assessment EVALUATION SHEET				
Course Outcome	**Experiment Name**	**Marks Obtained Out of 100**	**Average of each CO**	**CO Attained? (Y/N)**
20CSC409.1	1a. Install Virtualbox with different flavours of OS 1b. Install a C compiler in the virtual machine			
	1. Create instances and volume attachment in AWS			
20CSC409.2	1. Setup a Private Cloud Using Open Stack or Eucalyptus			
	2. Install Open Stack Object Storage -Swift in Ubuntu.			
	3. Implement OpenStack Nova Compute and Image services Glance.			
	4. Setup Openstack Using DevStack.			
20CSC409.3	1. Install a Single node hadoop cluster			
	2. Implement Map Reduce concept for word count program			
20CSC409.4	1. Implement VM Scheduling Using CloudSim			
20CSC409.5	1. Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories using Git Bash and Git Hub.			
	Over All Total			
Average Mark (Out of 100) – (A)				

Co Attainment (Internal Assessment) Evaluation				
Experiment		**Model Exam**		**CO Attained**
A	**B**	**C**	**D**	
Mark Obtained (100)	Assessment Weightage (75% of 100 Marks Obtained)	Mark Obtained (100)	Assessment Weightage 25% of Mark Obtained	**E=((B+D) *0.50)**

Measuring Tools for CO attainment (Components & Weight age)			
Internal Experiment - 75% Model Exam - 25%	50%	External University Exam	**50%**

Action Plan: (If CO Not Attained)				
CO	**Reason for not Attaining CO**	**Remedial Action Plan**	**Remarks**	**Signature**

Rubrics for assessment

Course Code (As per Curriculum)	:	**CS1718**
Course Code (As per NBA)	:	**20CSC409**
Course Name	:	**Cloud Computing Laboratory**

Rubrics

Category		Rubrics
Discipline (3)	Dress Code	1 Mark - Students must wear lab coat. 0 Mark - No lab coat.
	Note book	1 Mark - Students must bring their manual cum record note book. 0 Mark - No manual cum record note book.
	Punctuality	1 Mark - Students must be present inside the lab in time. 0 Mark - Not in time.
Preparation (2)	Prior knowledge on experiment	2 Marks - Students should write the program for the given experiment. 1 Mark – Partially written. 0 Mark –No idea about the experiment.
Performance (5)	Completion	3 Marks - Students should successfully complete the execution within the allotted hours. 1 Mark - Partially completed.
	Participation	2 Marks - Students should involve themselves while doing their experiments. 1 Mark - Involvement is less.
Viva-Voce (5)	Based on the performance, marks will be allotted.	
Record (5)	Completion	5 Marks - The students should submit their completed record before entering into next lab. 3 Marks - Late submission within a week. 1 Mark – Late submission after one week.

Ex. No: 1 a	Install Virtual box with different flavours of OS
Date:	

AIM

To install Guest OS on Virtual Machine using Oracle VirtualBox.

DESCRIPTION

VIRTUALIZATION

- Virtualization is the creation of virtual servers, infrastructures, devices and computing resources.
- Virtualization changes the hardware-software relations and is one of the foundational elements of cloud computing technology that helps utilize the capabilities of cloud computing to the full.
- Virtualization techniques allow companies to turn virtual their networks, storage, servers, data, desktops and applications.

Hypervisor or Virtual Machine Monitor (VMM)

A **hypervisor** or **virtual machine monitor** (**VMM**) is a piece of computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor is running one or more virtual machines is defined as a *host machine*. Each virtual machine is called a *guest machine*. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources.

Types of Virtualization

- **Operating System Level Virtualization:** Server virtualization method where the kernel of an operating system allows for multiple isolated user- space instances, instead of just one. Such instances (sometimes called **containers**, **software containers**, virtualization engines (VE), virtual private servers (VPS)) may look and feel like a real server from the point of view of its owners and users.

- **Platform / Hardware Virtualization:** Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine.
- **In hardware virtualization**, the host machine is the actual machine on which the virtualization takes place, and the guest machine is the virtual machine. The words *host* and *guest* are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine. Different types of hardware virtualization include:
    - **Full Virtualization:** Almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.
    - **Partial Virtualization:** Some but not all of the target environment is simulated. Some guest programs, therefore, may need modifications to run in this virtual environment.
    - **Para Virtualization:** A hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system.
- **Application Virtualization:** A software technology that encapsulates computer programs from the underlying operating system on which it is executed. A fully virtualized application is not installed in the traditional sense, although it is still executed as if it were.

PROCEDURAL STEPS

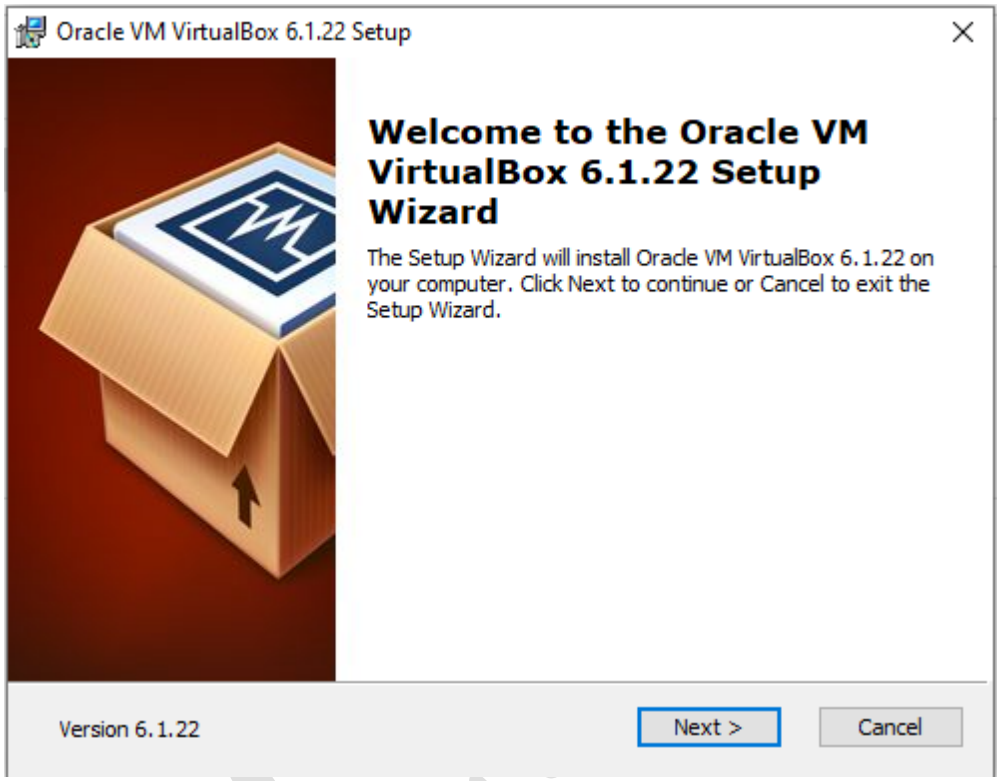
1. Download and Install Oracle Virtual Box latest version & Extension Package.

   Link 1: https://www.virtualbox.org/wiki/Downloads

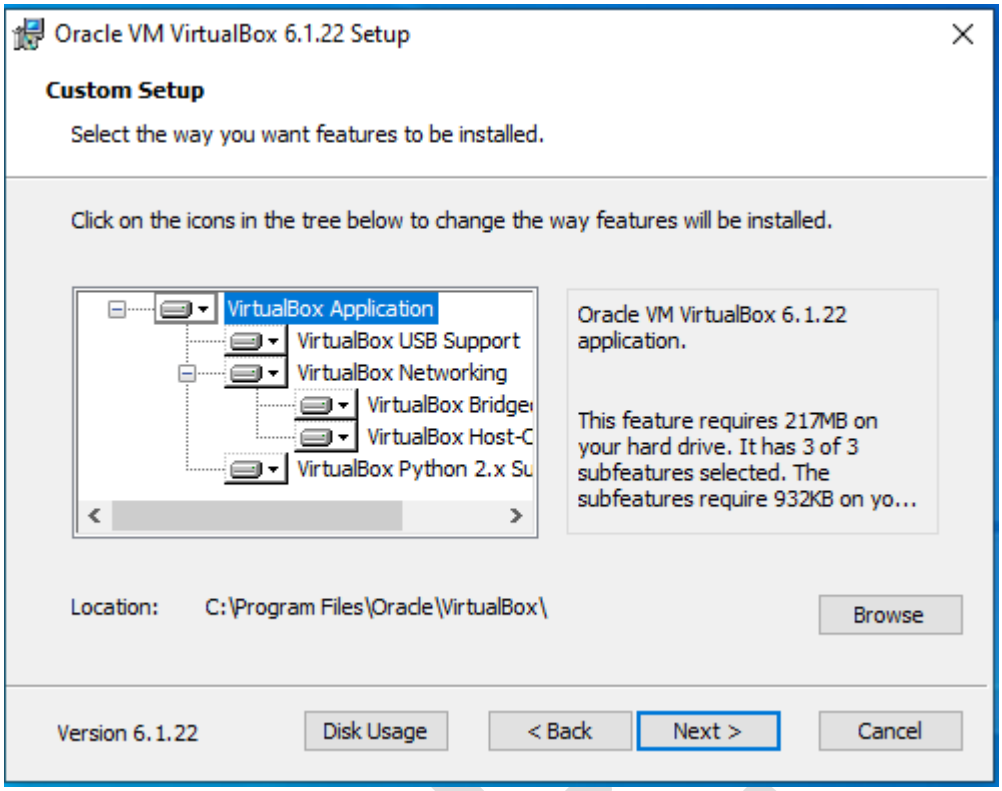
   Link 2: https://www.oracle.com/in/virtualization/technologies/vm/downloads/virtualbox-downloads.html

2. Download Ubuntu 14.4 OVA (Open Virtual Appliance)
   Link: https://www.osboxes.org/ubuntu/

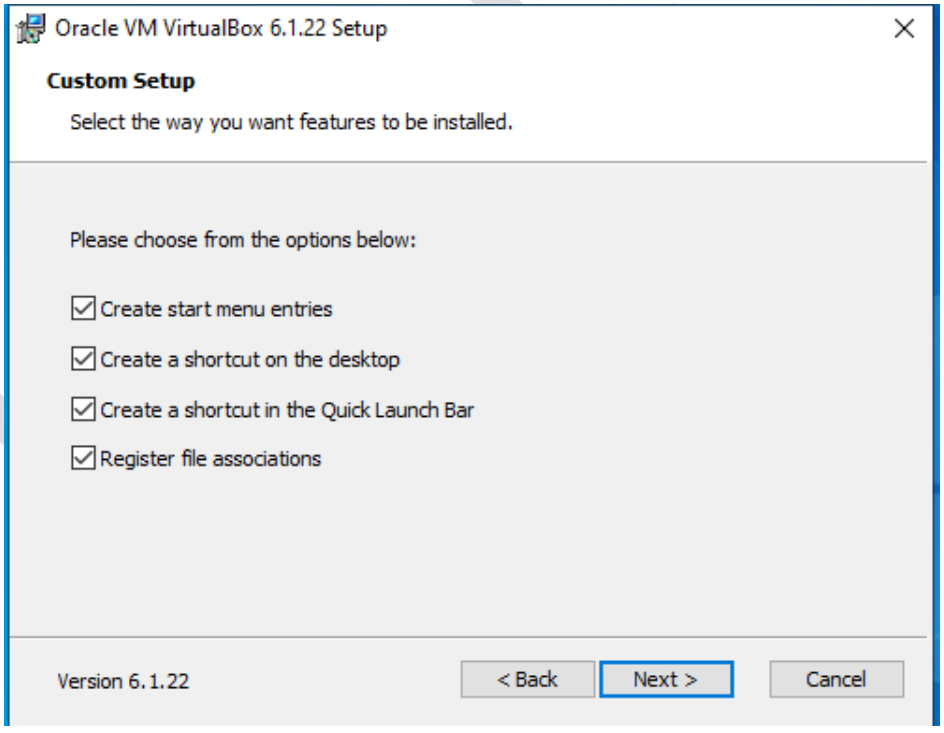
3. The files are downloaded in Local Machine → Click the Oracle VM VirtualBox 6.0.8 & Setup Wizard Move to run time environment (Open Terminal)



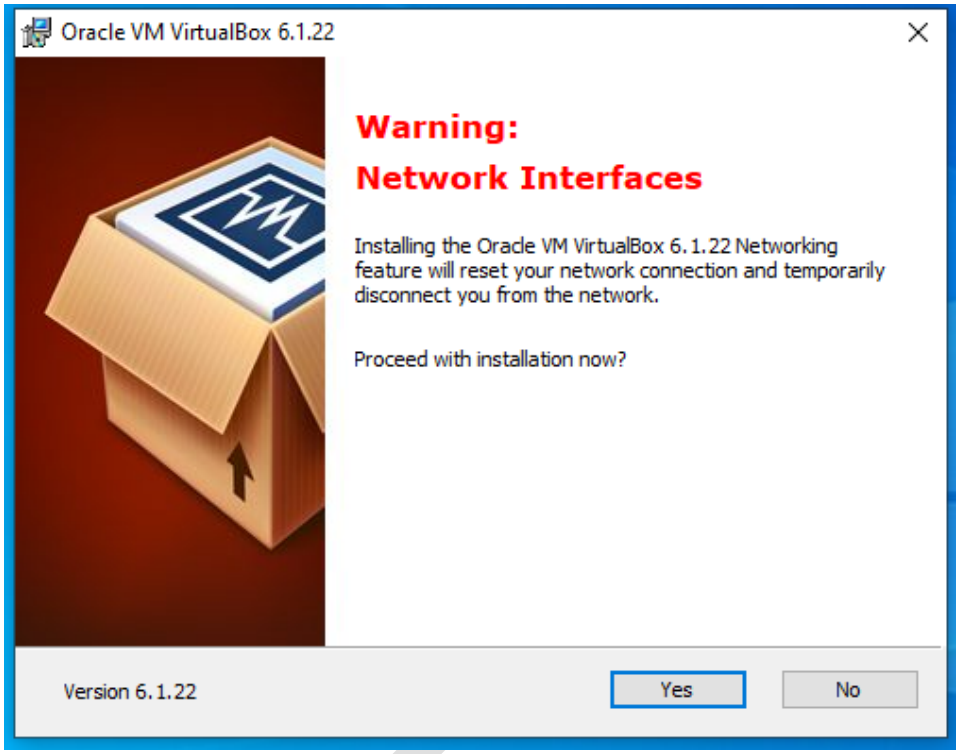
4. Custom Setup → Check VirtualBox Application



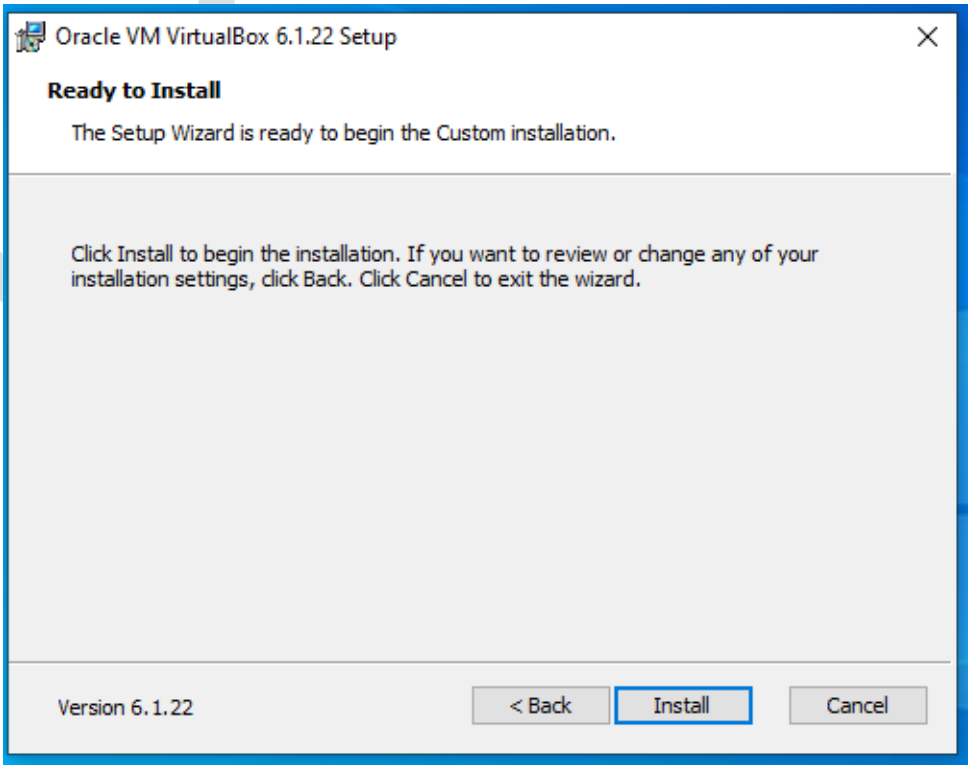
5. Custom Setup → Select the features to be installed



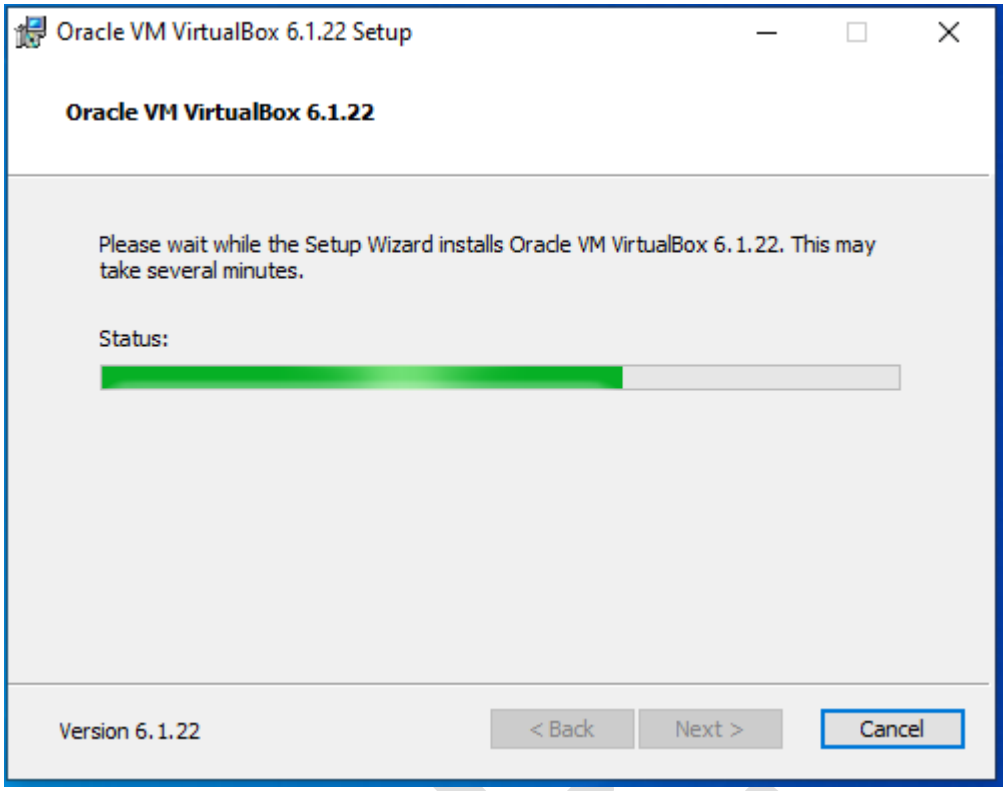
6. Warning: Network Interfaces → Click ‘Yes’ & Proceed to install



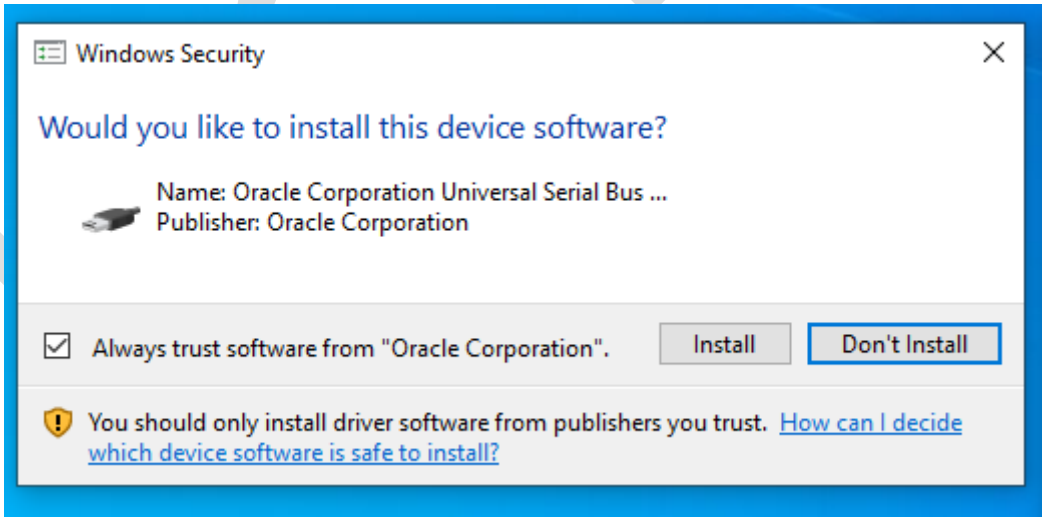
7. Ready to install → Click 'Install'



8. Oracle VM VirtualBox 6.1.22 installing

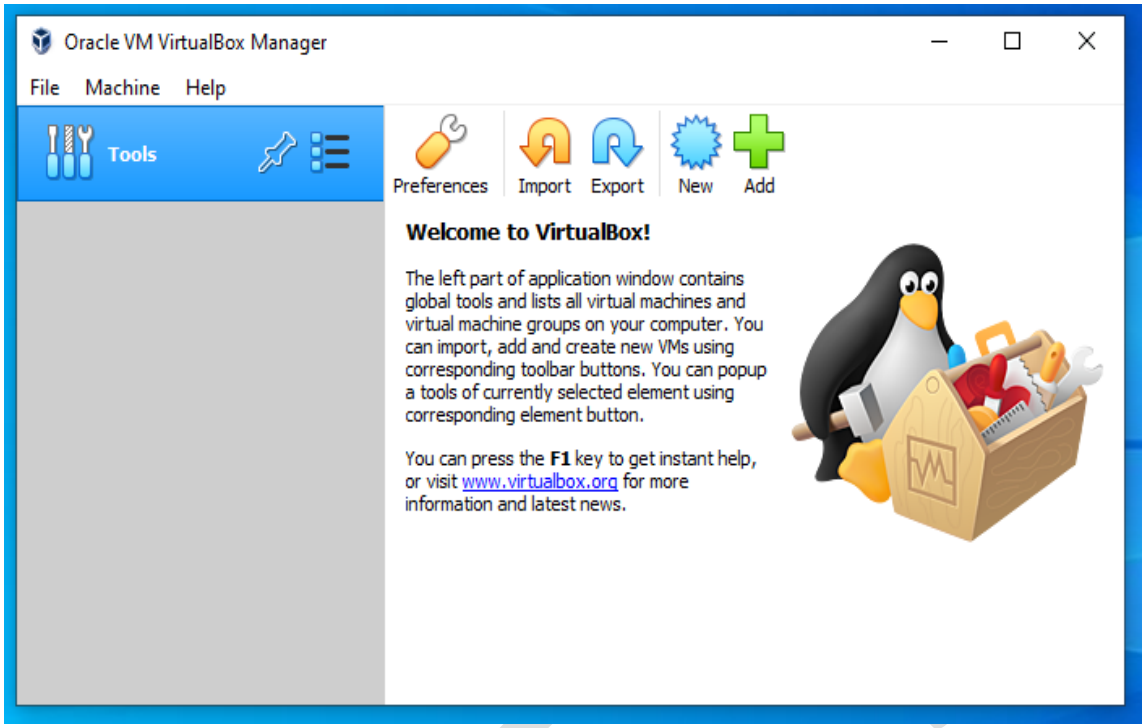


9. To install device software → Click ‘Install’

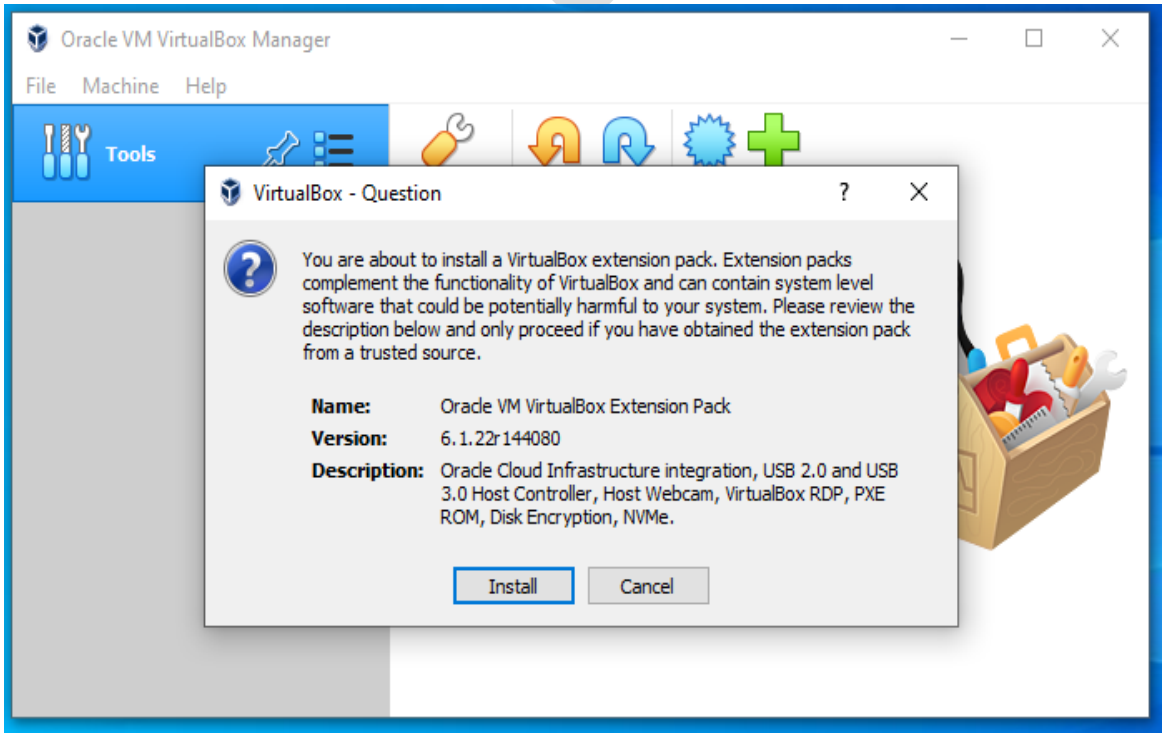


10. Oracle VM VirtualBox 6.1.22 installation is complete → Click ‘Finish’

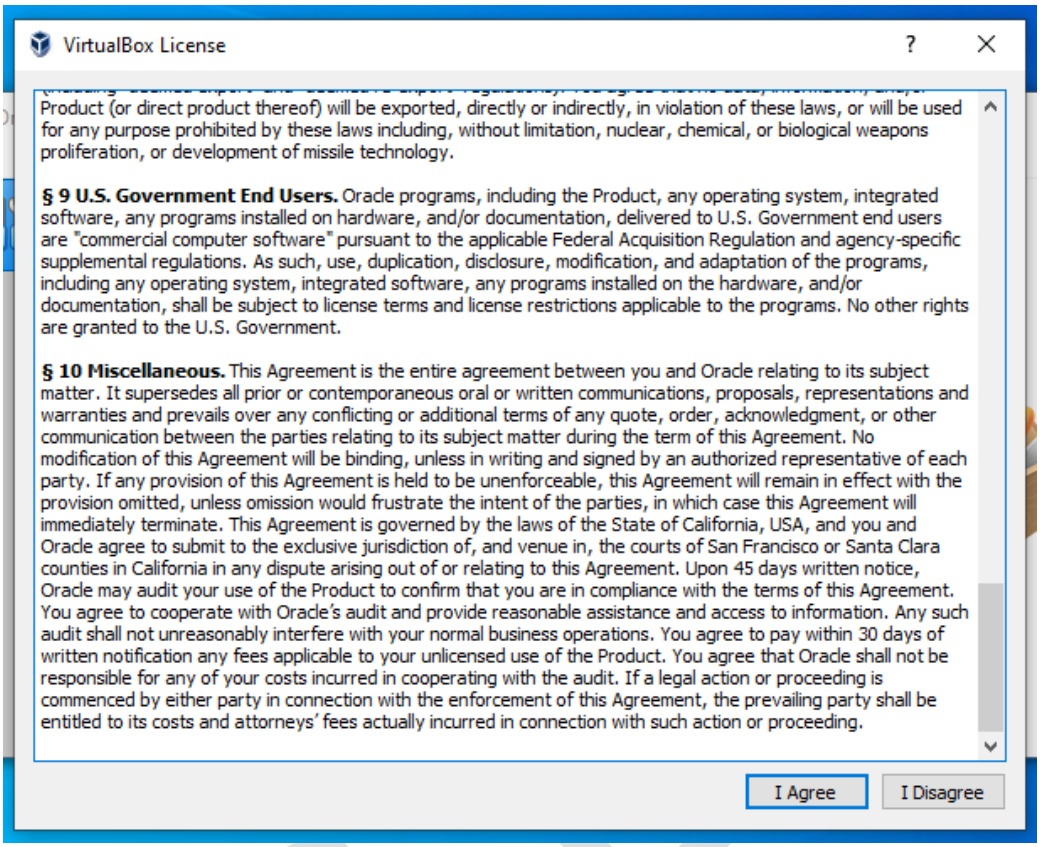
11. Import the Oracle VM Virtual Extension Pack into Oracle VirtualBox



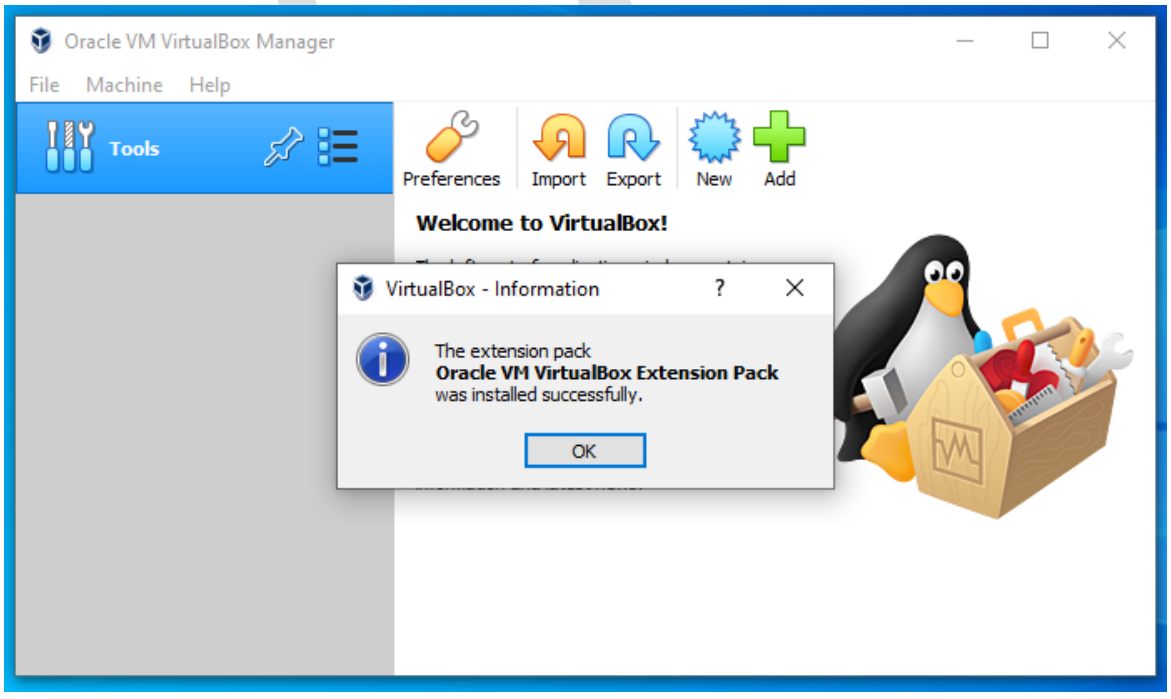
12. Click ‘Install’ Oracle VM Virtual Extension Pack



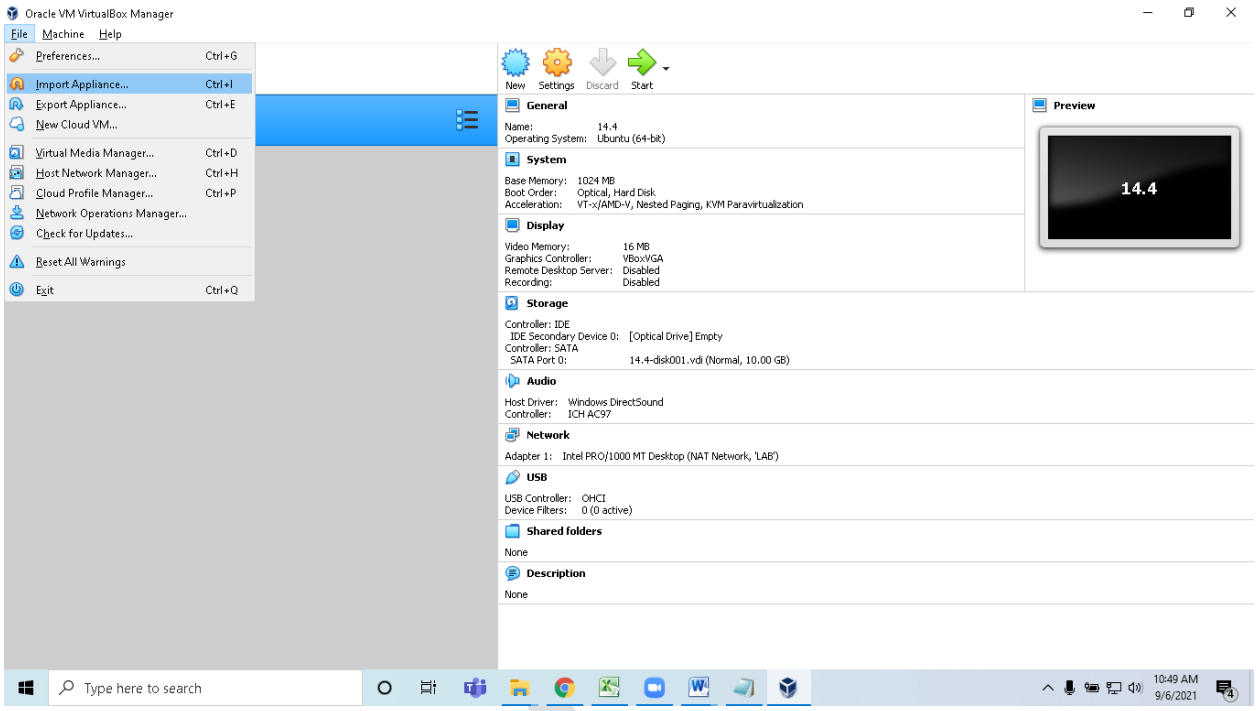
13. VirtualBox License → Click ‘I Agree’

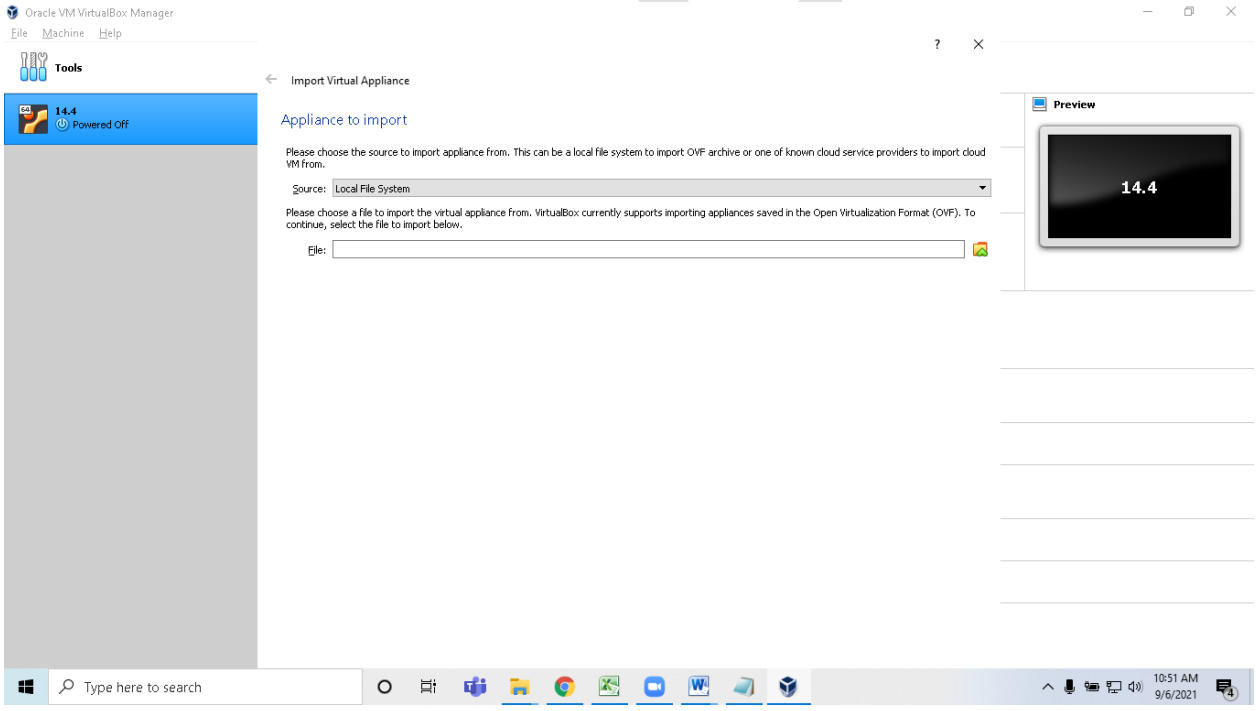


14. Extension Pack installation successful → Click ‘OK’

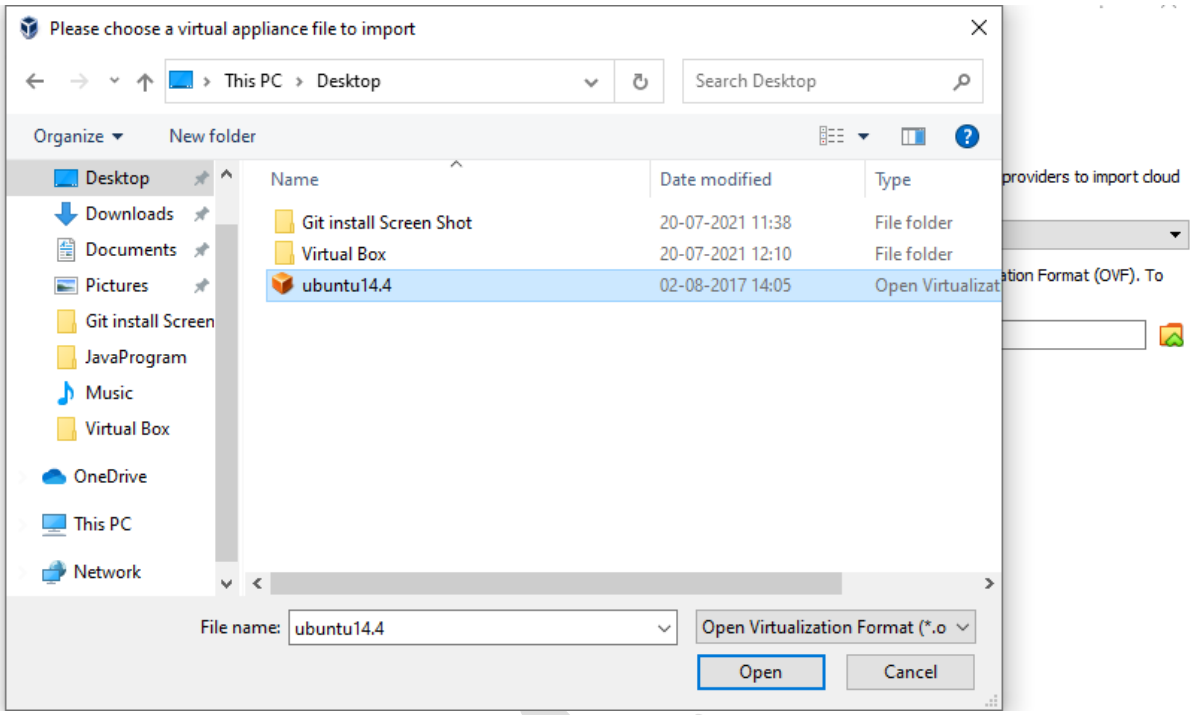


15. To install Virtual Machine: File → Import Appliance

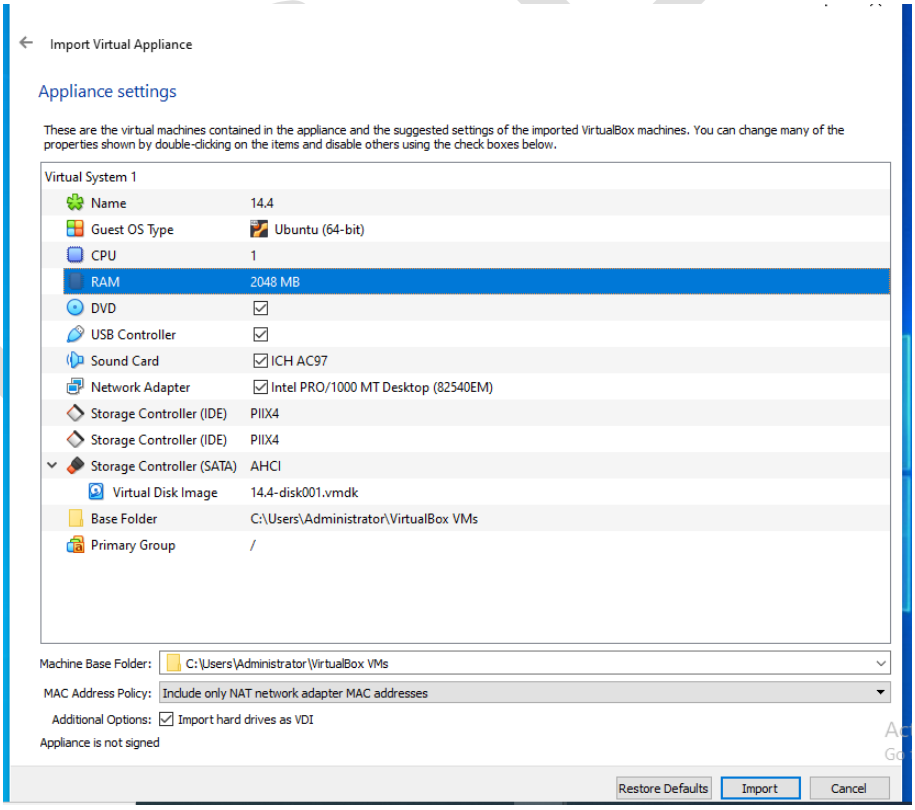




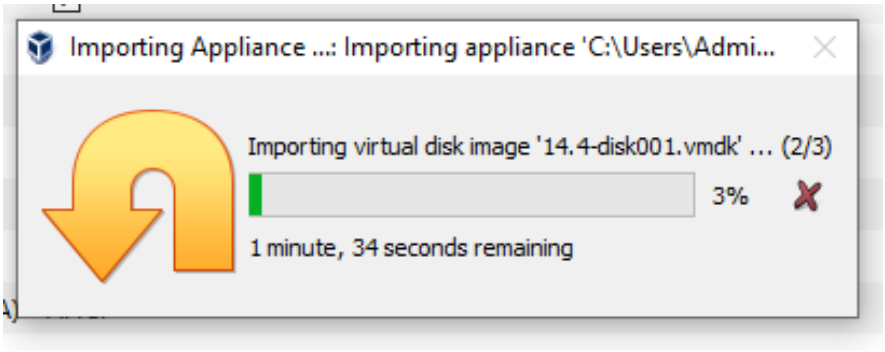
16. Select Ubuntu 14.4 OVA from directory



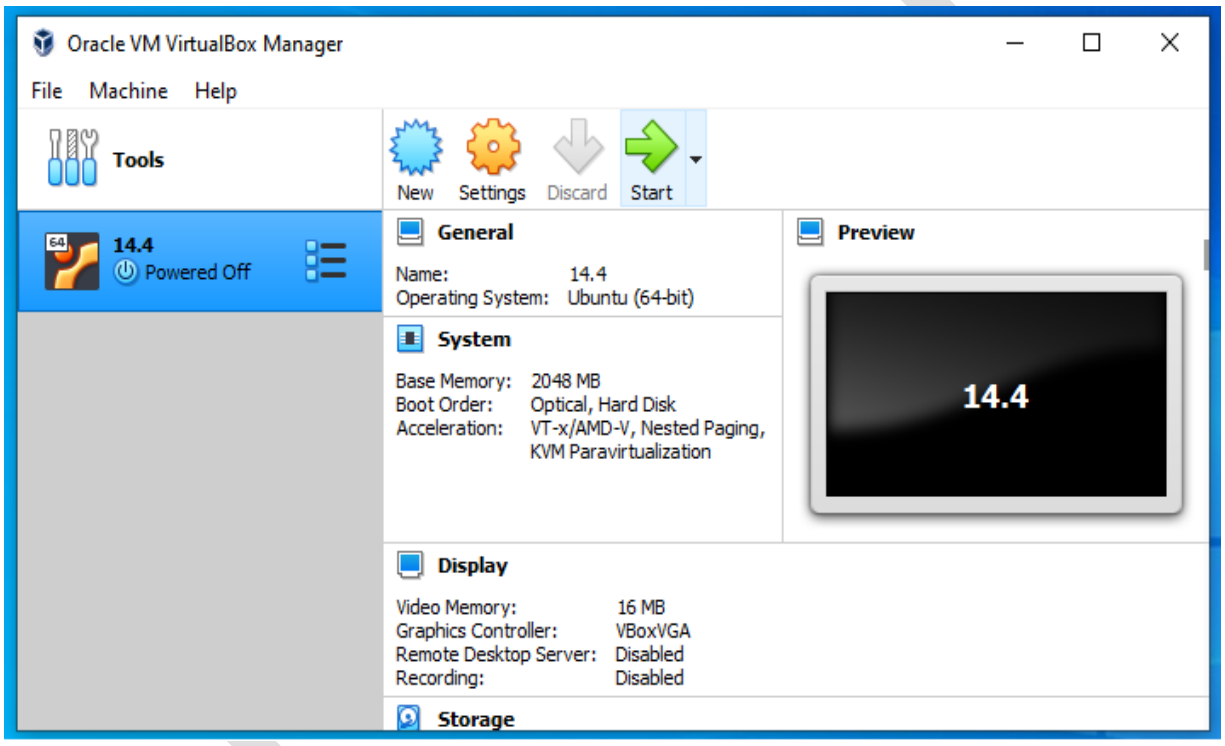
17. Appliance Settings → Choose ‘RAM’ Size → Click ‘Import’



18. Importing the Virtual Disk Image



19. Guest OS ‘Ubuntu 14.4’ is installed successfully and Click ‘Start’ button to lunch the virtual machine



20. Login to Ubuntu 14.4

Login Details:

User name: hadoop

Password: Test1234

VIVA QUESTIONS

1. What is virtualization in cloud computing?

2. What is Host OS in Virtualization?

3. What is Guest OS in Virtualization?

4. List some Virtual Machine Software.

5. Name the Virtual Machine Software used in our lab exercise.

RESULT

Thus, the Guest OS is installed on Virtual Machine using Oracle VirtualBox.

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

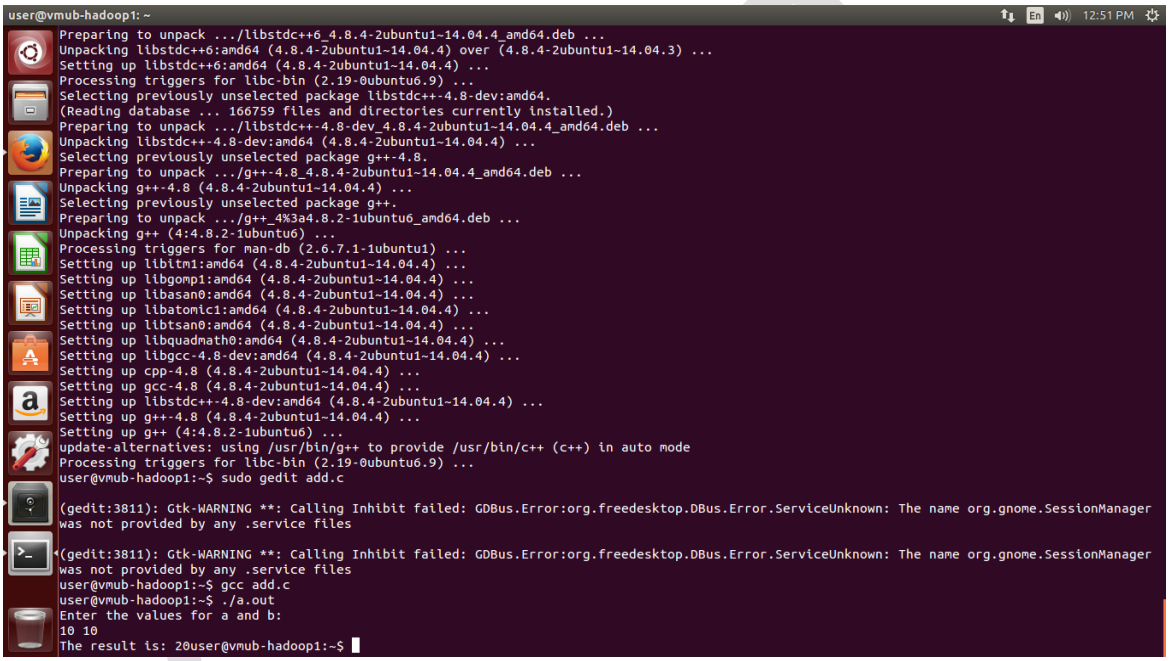
Ex. No: 1b	Installing C Complier on Guest OS
Date:	

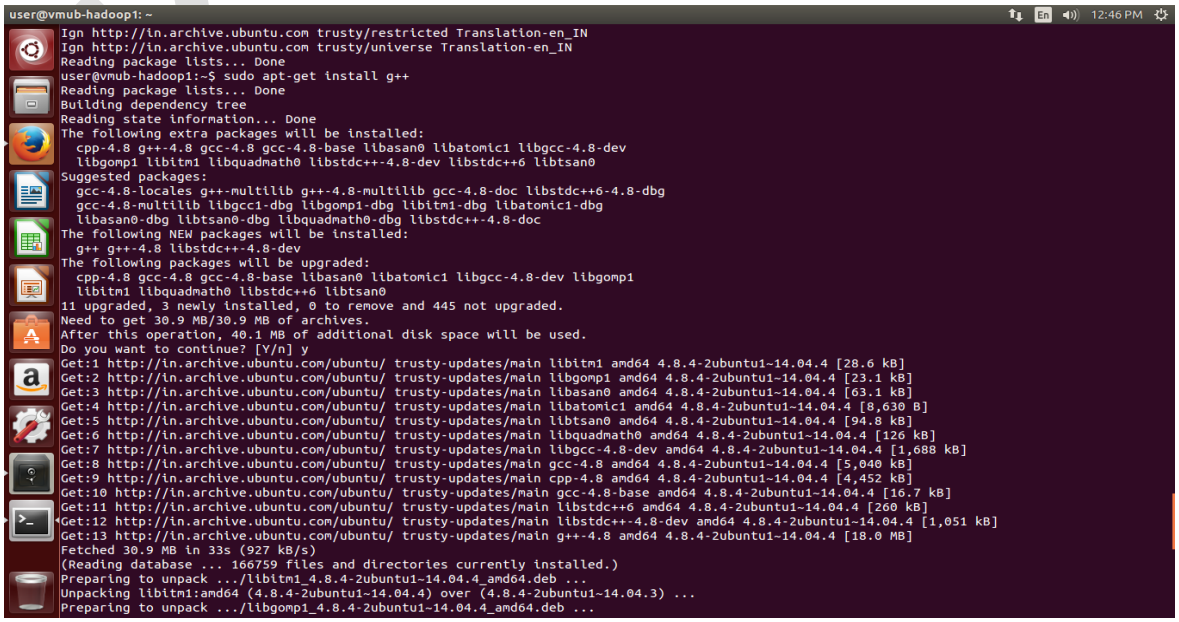
AIM

To install C Complier on Guest OS ‘Ubuntu 14.4’.

PROCEDURAL STEPS

1. To download package information from all configured sources → **$ sudo apt-get update**
2. To install C Compiler on Ubuntu 14.4 → **$ sudo apt-get install gcc**
3. To install C++ Compiler on Ubuntu 14.4 → **$ sudo apt-get install g++**



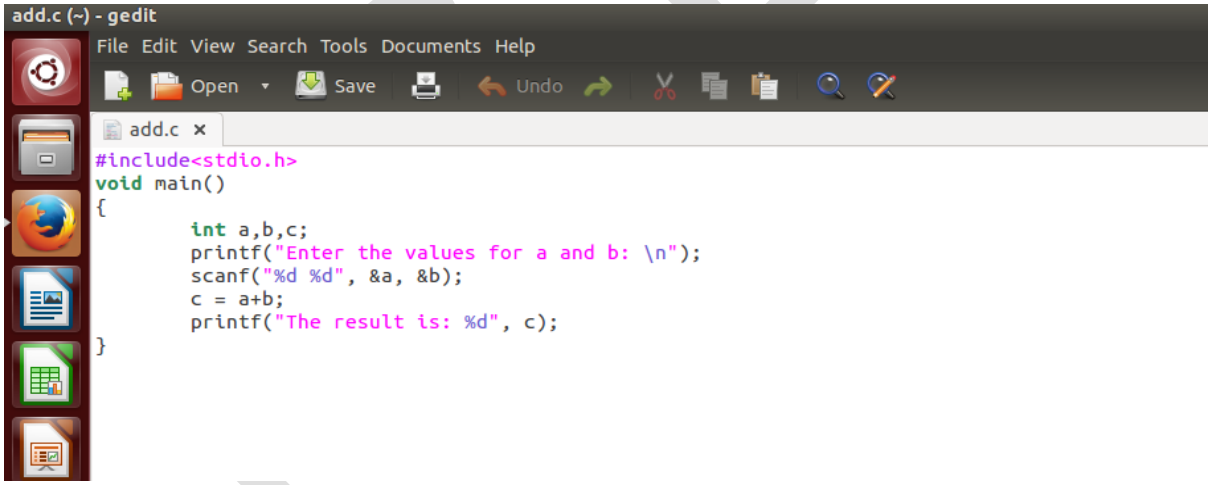


```
user@vmub-hadoop1: ~
user@vmub-hadoop1:~$ sudo apt-get install gcc
[sudo] password for user:
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 296 not upgraded.
user@vmub-hadoop1:~$ sudo apt install g++
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  g++-4.8 libstdc++-4.8-dev
Suggested packages:
  g++-multilib g++-4.8-multilib gcc-4.8-doc libstdc++6-4.8-dbg
  libstdc++-4.8-doc
The following NEW packages will be installed:
  g++ g++-4.8 libstdc++-4.8-dev
0 upgraded, 3 newly installed, 0 to remove and 296 not upgraded.
Need to get 19.2 MB of archives.
After this operation, 40.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

4. Create a file to write C program → **$ sudo gedit <<*file_name*>>.c**

   To compile C program → **$ gcc <<*file_name*>>.c**

   To run C program → **$ ./a.out**




```
user@vmub-hadoop1:~$ sudo gedit add.c

(gedit:3811): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.free
was not provided by any .service files

(gedit:3811): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.free
was not provided by any .service files
user@vmub-hadoop1:~$ gcc add.c
user@vmub-hadoop1:~$ ./a.out
Enter the values for a and b:
10 10
The result is: 20user@vmub-hadoop1:~$
```

arm.c (~) - gedit

Open Save Undo

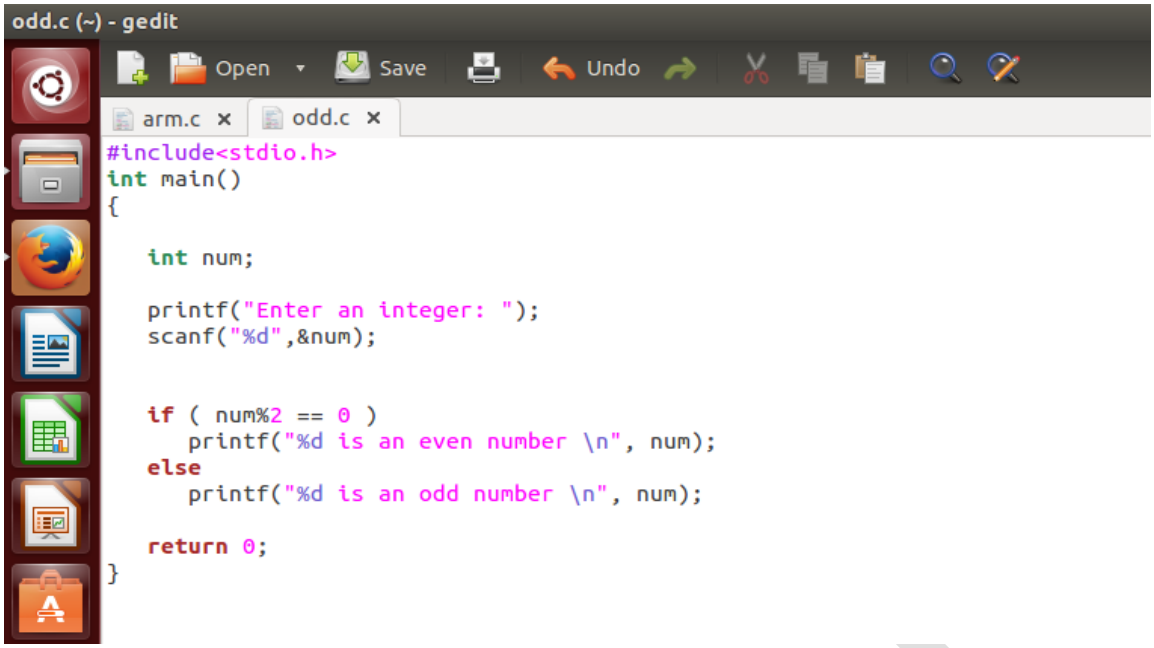
arm.c × odd.c ×

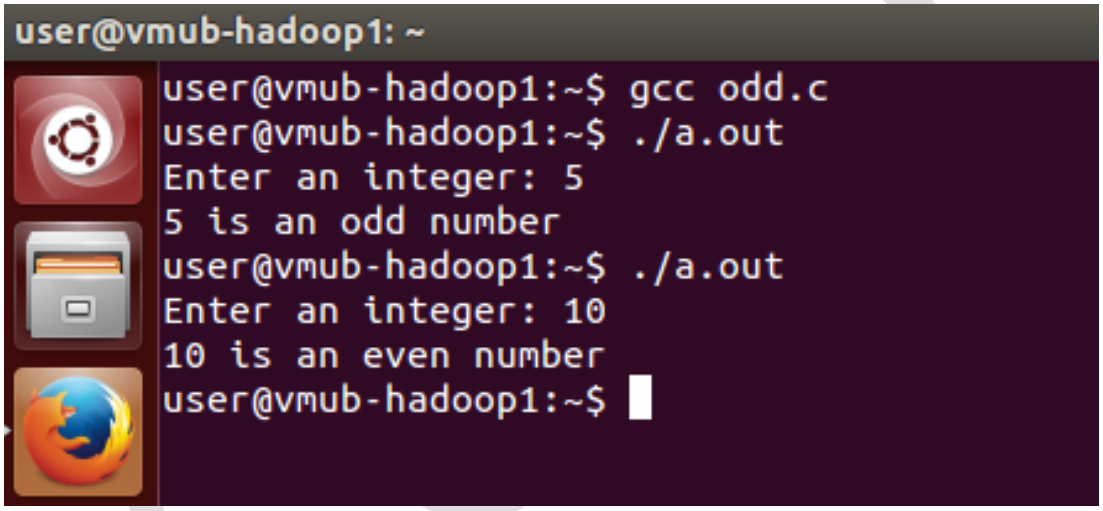
```c
#include<stdio.h>

int main()
{
        int n,r,sum=0,temp;
        printf("enter the number = ");
        scanf("%d",&n);
        temp=n;
        while(n>0)
        {
                r=n%10;
                sum=sum+(r*r*r);
                n=n/10;
        }
        if(temp==sum)
                printf("armstrong  number \n");
        else
                printf("not armstrong number \n");
        return 0;
}
```

user@vmub-hadoop1: ~

```
user@vmub-hadoop1:~$ gcc arm.c
user@vmub-hadoop1:~$ ./a.out
enter the number = 153
armstrong  number
user@vmub-hadoop1:~$ ./a.out
enter the number = 150
not armstrong number
user@vmub-hadoop1:~$
```





VIVA QUESTIONS

1. What is Parallel Computing?

2. What is Distributed System?

3. List the compilers to run C program.

4. Name the command to compile C program in Linux.

RESULT

Thus, the C Complier is installed on Guest OS ‘Ubuntu 14.4’ and simple programs are executed.

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

Ex. No: 2	Create instances and volume attachment in AWS
Date:	

AIM

To Create instances and volume attachment in AWS

DESCRIPTION

AMAZON WEB SERVICES (AWS)

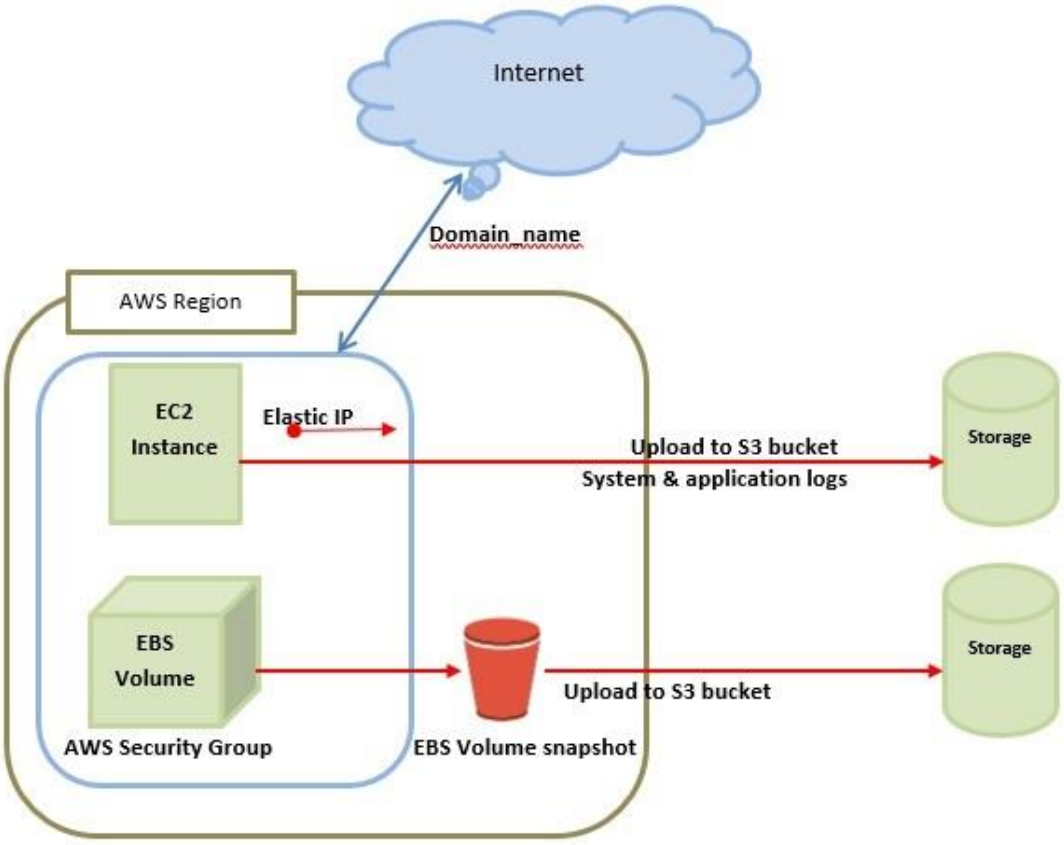
Amazon Web Services (AWS) is Amazon's cloud web hosting platform that offers flexible, reliable, scalable, easy-to-use, and cost-effective solutions. This tutorial covers various important topics illustrating how AWS works and how it is beneficial to run your website on Amazon Web Services.

AWS Basic Architecture

AWS EC2, where **EC2** stands for Elastic Compute Cloud. EC2 allow users to use virtual machines of different configurations as per their requirement. It allows various configuration options, mapping of individual server, various pricing options, etc. We will discuss these in detail in AWS Products section.

AWS S3

Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services that provides object storage through a web service interface. Amazon S3 uses the same scalable storage



Note − In the above diagram **S3** stands for Simple Storage Service. It allows the users to store and retrieve various types of data using API calls. It doesn't contain any computing element. We will discuss this topic in detail in AWS products section.

PROCEDURAL STEPS

1.To create instances and attach volumes in AWS, you can follow these steps:

2.Sign in to the AWS Management Console at https://console.aws.amazon.com/.

3.Open the EC2 service by searching for "EC2" in the AWS services search bar and selecting it.

4.In the EC2 Dashboard, click on "Instances" in the left navigation pane.

5.Click on the "Launch Instance" button to start the instance creation wizard.

6.Choose an Amazon Machine Image (AMI) that suits your requirements. This is the base operating system for your instance.

7.Select an instance type based on the resources and performance you need.

8.Configure the instance details such as the number of instances, network settings, security groups, and IAM roles. Customize these settings according to your specific needs.

9.Add storage by specifying the size and type of the root volume or by adding additional volumes as needed. You can also configure encryption and other settings for the volumes.

10.Configure security groups to control inbound and outbound traffic to your instance.

11.Review the instance details and configuration, and make any necessary changes.

12.Finally, click on the "Launch" button. You will be prompted to select or create a key pair to securely connect to your instance.

13.Once the instance is launched, you can find it listed in the EC2 Dashboard under "Instances."

14.To attach a volume to your instance:

15.In the EC2 Dashboard, click on "Volumes" in the left navigation pane.

16.Click on the "Create Volume" button to start the volume creation wizard.

17. Select the desired volume type, size, and availability zone.

18.Click on the "Create" button to create the volume.

19.Once the volume is created, select it from the Volumes list.

20. Click on the "Actions" button and choose "Attach Volume."

21. In the Attach Volume dialog box, select the instance to which you want to attach the volume.

22.Specify the device name (e.g., /dev/sdf) for the attachment.

23.Click on the "Attach" button to attach the volume to the instance.

24.The volume will now appear as "in-use" in the Volumes list, indicating a successful attachment.

25. You have now created an instance and attached a volume in AWS.

VIVA QUESTIONS

1. What is an Instance in AWS?

2. What is Platform as a Service (PaaS)?

3. List some PaaS examples.

4. What is Elastic Bean Stalk?

RESULT

Thus, the Instance is created and the volume is attached in the AWS console

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

Ex. No: 3	Setup a Private Cloud Using Open Stack or Eucalyptus
Date:	

AIM

To setup a private cloud using open stack or eucalyptus.

PROCEDURAL STEPS

Step 1: Hardware and Network Requirements

Ensure that you have the necessary hardware resources and network infrastructure to support your private cloud deployment. This typically includes servers, storage devices, network switches, and sufficient network bandwidth.

Step 2: Install a Base Operating System

Choose a Linux distribution (e.g., Ubuntu, CentOS) and install it on the servers that will be part of your OpenStack deployment. Follow the installation instructions provided by the distribution.

Step 3: Configure Network Settings

Ensure that each server has a unique hostname and a static IP address. Set up DNS resolution so that each server can resolve the others' hostnames.

Step 4: Install OpenStack Packages

Update your system packages and install the necessary OpenStack packages. The exact packages and their installation process may vary depending on the Linux distribution you're using. Refer to the OpenStack documentation for the specific packages and installation instructions.

Step 5: Configure the Identity Service (Keystone)

Configure the Identity service (Keystone) to authenticate and manage users, projects, and roles within your OpenStack cloud. This involves setting up a database, configuring the Keystone service, and creating initial users and projects.

Step 6: Configure the Image Service (Glance)

Configure the Image service (Glance) to store and manage virtual machine images. This involves setting up a storage backend, configuring Glance services, and adding initial images to the image catalog.

Step 7: Configure the Compute Service (Nova)

Configure the Compute service (Nova) to manage and orchestrate virtual machines. This includes setting up hypervisor drivers, configuring Nova services, and defining flavors, which specify the compute resources available to instances.

Step 8: Configure the Networking Service (Neutron)

Configure the Networking service (Neutron) to provide network connectivity to instances. This involves setting up network plugins, creating networks, subnets, and routers, and configuring security groups and floating IPs.

Step 9: Configure the Block Storage Service (Cinder)

Configure the Block Storage service (Cinder) to provide persistent storage to instances. This includes setting up storage backends, configuring Cinder services, and creating volumes and volume types.

Step 10: Configure the Dashboard (Horizon) (Optional)

Optionally, configure the Horizon dashboard, which provides a graphical user interface for managing your OpenStack cloud. This involves setting up the dashboard, configuring authentication, and customizing the interface if desired.

Step 11: Test and Verify

After completing the configuration of each OpenStack component, test the functionality to ensure that everything is working as expected. Create instances, assign floating IPs, attach volumes, and test network connectivity.

Step 12: Secure and Monitor

Implement security best practices to secure your private cloud environment. Set up monitoring tools to track the health and performance of your OpenStack infrastructure.

Remember that setting up a private cloud can be complex, and the above steps provide a general outline. It is recommended to refer to the official OpenStack documentation and consult relevant guides or tutorials specific to your chosen Linux distribution for detailed instructions and troubleshooting information

VIVA QUESTIONS

1. What is a Private Cloud?

2. What is Open Stack?

3. What are the Open Stack components?

4. What are the features of Open Stack

RESULT

Thus, a Private Cloud is setup using Open Stack

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

Ex. No: 4 Date:	Install Open Stack Object Storage -Swift in Ubuntu

AIM

To Install open stack object storage – swift in ubuntu.

PROCEDURAL STEPS

Step 1: Update System Open a terminal and run the following command to update the system packages:

sudo apt update

sudo apt upgrade

```
sudo apt update
sudo apt upgrade
```

Step 2: Install Swift Dependencies Swift requires certain dependencies to be installed on your Ubuntu system. Use the following command to install them:

sudo apt install swift swift-proxy swift-object swift-account swift-container

```
sudo apt install swift swift-proxy swift-object swift-account swift-containe
```

Step 3: Configure Swift Next, you need to configure the Swift object storage system. The configuration files are located in the /etc/swift directory. Open the /etc/swift/swift.conf file in a text editor and modify the following values:

[swift-hash]

Set the value to match your drive

swift_hash_path_suffix = RANDOM_HASH

```
[swift-hash]
# Set the value to match your drive
swift_hash_path_suffix = RANDOM_HASH
```

Replace RANDOM_HASH with a randomly generated hash value.

Step 4: Create Object Storage Directories Swift requires specific directories to store its data. Run the following commands to create these directories:

sudo mkdir /var/cache/swift

sudo mkdir /var/run/swift

sudo chown -R swift:swift /var/cache/swift /var/run/swift

```
sudo mkdir /var/cache/swift
sudo mkdir /var/run/swift
sudo chown -R swift:swift /var/cache/swift /var/run/swift
```

Step 5: Start Swift Services To start the Swift services, run the following commands:

sudo systemctl start swift-account

sudo systemctl start swift-container

sudo systemctl start swift-object

sudo systemctl start swift-proxy

```
sudo systemctl start swift-account
sudo systemctl start swift-container
sudo systemctl start swift-object
sudo systemctl start swift-proxy
```

Step 6: Enable Swift Services on Boot To ensure that the Swift services start automatically on system boot, run the following commands:

sudo systemctl enable swift-account

sudo systemctl enable swift-container

sudo systemctl enable swift-object

sudo systemctl enable swift-proxy

```
sudo systemctl enable swift-account
sudo systemctl enable swift-container
sudo systemctl enable swift-object
sudo systemctl enable swift-proxy
```

Step 7: Verify Installation To verify that Swift is installed and running correctly, you can check the service status:

sudo systemctl status swift-account

sudo systemctl status swift-container

sudo systemctl status swift-object

sudo systemctl status swift-proxy

```
sudo systemctl status swift-account
sudo systemctl status swift-container
sudo systemctl status swift-object
sudo systemctl status swift-proxy
```

If all the services are active and running, then Swift is successfully installed on your Ubuntu system.

That's it! You have successfully installed Object Storage Swift on Ubuntu. You can now use Swift to store and retrieve objects as needed.

VIVA QUESTIONS

1. What is Swift in Ubuntu?

2. State the purpose of object storage.

3.What is the difference between Open stack swift and and Swift stack?

4.What is the use of Swiftstack?

RESULT

Thus, the Open Stack Object Storage Swift Installed in Ubuntu..

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

Ex. No: 5	Implement OpenStack Nova – Compute and Image services Glance
Date:	

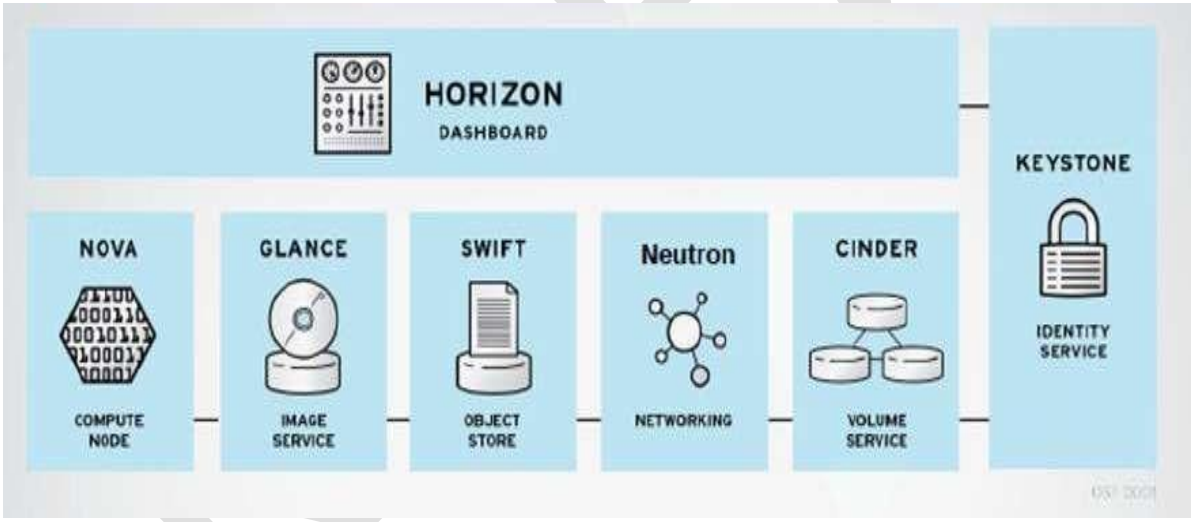
AIM

To Implement OpenStack Nova – Compute and Image services –Glance

DESCRIPTION

- OpenStack was introduced by Rackspace and NASA in July 2010.
- OpenStack is an Infrastructure as a Service known as Cloud Operating System, that take resources such as Compute, Storage, Network and Virtualization Technologies and control those resources at a data center level
- The project is building an open-source community to share resources and technologies with the goal of creating a massively scalable and secure cloud infrastructure.
- The software is open source and limited to just open-source APIs such as Amazon.

OpenStack Architecture



- It is modular architecture
- Designed to easily scale out
- Based on (growing) set of core services

OpenStack Major Components

1. Keystone
2. Nova
3. Glance
4. Swift

5. Quantum
6. Cinder

KEYSTONE

- Identity service
- Common authorization framework
- Manage users, tenants and roles
- Pluggable back-ends (SQL,PAM,LDAP, IDM etc)

NOVA

- Core compute service comprised of
    - Compute Nodes: Hypervisors that run virtual machines
- Supports multiple hypervisors KVM, Xen, LXC, Hyper-V and ESX
    - Distributed controllers that handle scheduling, API calls, etc.
- Native OpenStack API and Amazon EC2 compatible API

GLANCE

- Image service
- Stores and retrieves disk images (Virtual machine templates)
- Supports RAW, QCOW, VHD, ISO, OVF & AMI/AKI
- Back-end Storage: File System, Swift, Gluster, Amazon S3

SWIFT

- Object Storage service
- Modeled after Amazon‘s Service
- Provides simple service for storing and retrieving arbitrary data
- Native API and S3 compatible API

NEUTRON

- Network service
- Provides framework for Software Defined Network
- Plugin architecture
    - Allows integration of hardware and software-based network solutions
    - Open vSwitch, Cisco UCS, Standard Linux Bridge, NiCira NVP

CINDER

- Block Storage (Volume) service
- Provides block storage for Virtual Machines (persistent disks)

- Like Amazon EBS service
- Plugin architecture for vendor extensions
- NetApp driver for cinder

HORIZON

- Dashboard
- Provides simple self-service UI for end-users
- Basic cloud administrator functions
    - Define users, tenants, and quotas
    - No infrastructure management

HEAT OpenStack Orchestration

- Provides template driven cloud application orchestration
- Modeled after AWS Cloud Formation
- Targeted to provide advanced functionality such as high availability and auto scaling
- Introduced by Redhat

CEILOMETER

- OpenStack Monitoring and Metering
- **Goal:** To Provide a single infrastructure to collect measurements from an entire OpenStack Infrastructure; Eliminate need for multiple agents attaching to multiple OpenStack Projects
- Primary targets metering and monitoring: Provided extensibility

PROCEDURAL STEPS:

Prepare the Environment:

Step 1: Set up a physical or virtual machine to serve as the compute node.

Install a supported operating system (typically a Linux distribution).

Ensure the machine meets the minimum requirements for running OpenStack Nova-Compute.

Step2 : Install and Configure the Compute Node:

Step 3: Update the system packages and repositories.

Step 4: Install the necessary dependencies, such as Python and other required packages.

Step 5: Install and configure the hypervisor software you plan to use (e.g., KVM, VMware).

Step 6: Install the Nova-Compute service and its dependencies using the package manager (e.g., apt, yum).

Step 7: Configure the Nova-Compute service by editing the configuration file (/etc/nova/nova.conf). Set appropriate values for options like authentication, messaging, networking, and hypervisor-related settings.

Step 8: Optionally, configure additional features like live migration, GPU passthrough, etc., as per your requirements.

Configure Networking:

Step 1: Ensure the compute node has network connectivity to the OpenStack controller node and other services.

Step 2: Configure the network settings in the Nova-Compute configuration file (/etc/nova/nova.conf). Specify the appropriate network provider, the URL of the message queue service, authentication credentials, etc.

Step 3: Enable and Start Nova-Compute Service:

Step 4: Enable the Nova-Compute service to start automatically at boot time.

Step 5: Start the Nova-Compute service using the appropriate service manager for your operating system (e.g., systemctl, service).

Verify Compute Node Registration:

Step 1: On the OpenStack controller node, use the OpenStack command-line client or API to verify if the compute node is registered and recognized by the OpenStack environment.

Step 2: Validate the connectivity between the controller node and the compute node.

Step 3: Confirm that the hypervisor and other services on the compute node are running without errors.

Test Instance Launch:

Step 1: Use the OpenStack dashboard (Horizon) or command-line client to launch a test instance on the compute node.

Step 2: Monitor the instance creation process, ensuring that it is scheduled to run on the compute node.

Step 3: Verify that the instance becomes active and is accessible over the network.

By following these steps, you should be able to implement and configure OpenStack Nova-Compute successfully. Remember that this is a simplified overview, and you may need to consult the OpenStack documentation for more detailed information and troubleshooting guidance specific to your deployment scenario

To implement OpenStack Glance (Image Service), you need to follow these step-by-step procedures:

Set up the environment:

Install and configure the database:

Step 1 : Install and configure the database server (e.g., MySQL, MariaDB).

Step 2 : Create a database and database user for Nova and Glance services.

Install and configure message queue service:

Step 1 : Install RabbitMQ, which is the recommended message queue service for OpenStack.

Step 2 : Configure RabbitMQ to enable communication between OpenStack services.

Install and configure Identity service (Keystone):

Step 1: Install Keystone and configure the database and RabbitMQ options in the Keystone configuration file.

Step 2: Configure the admin token and endpoint URLs.

Install and configure Compute service (Nova):

Step 1: Install Nova Compute and its dependencies.

Step 2: Configure the database and RabbitMQ options in the Nova configuration file.

Step 3: Configure authentication and authorization settings in the Nova configuration file.

Step 4 : Configure the placement service (used for resource tracking) in the Nova configuration file.

Step 5: Start and enable the Nova Compute service.

Install and configure Image service (Glance):

Step 1: Install Glance and its dependencies.

Step 2: Configure the database and RabbitMQ options in the Glance configuration file.

Step 3: Configure the image storage backend (e.g., local file system, Swift, Ceph) in the Glance configuration file.

Step 4: Start and enable the Glance API and Glance Registry services.

Step 5: Verify the installation:

Step 6: Use the OpenStack command-line client (e.g., openstack) or API to verify that Nova and Glance services are running correctly.

Step 7: Test creating and managing instances using the Nova Compute service.

Step 8: Test uploading and retrieving images using the Glance Image service.

VIVA QUESTIONS

1. What is Nova ?

2. List the key components of OpenStack.

3. List the hypervisors supported by OpenStack.

4. What is Glance ?

RESULT

Thus, the OpenStack Nova – Compute and Image services –Glance is implemented

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

Ex. No: 6	Setup Openstack Using DevStack
Date:	

AIM

To Setup Openstack Using DevStack

PROCEDURAL STEPS:

Step 1: Set up a compatible operating system: DevStack officially supports Ubuntu, so start by installing Ubuntu Server on your machine. Make sure you have a clean installation without any conflicting services or configurations.

Step 2: Update the system: Run the following commands to update your Ubuntu system:

sudo apt update

sudo apt upgrade

```
sudo apt update
sudo apt upgrade
```

Step 3: Install Git: DevStack requires Git for source code retrieval and version control. Install Git by running the following command:

sudo apt install git

```
sudo apt install git
```

Step 4: Clone DevStack: Use Git to clone the DevStack repository:

git clone https://github.com/openstack-dev/devstack.git

```
git clone https://github.com/openstack-dev/devstack.git
```

Step 5: Customize DevStack configuration: Navigate to the DevStack directory and create a local.conf file to specify your OpenStack configuration options. For example, you can set the password, define the network, and enable/disable services. DevStack provides a sample configuration file (samples/local.conf) that you can use as a starting point. Copy it to your local.conf file:

cd devstack

cp samples/local.conf .

```
cd devstack
cp samples/local.conf .
```

Step 6: Modify local.conf: Open the local.conf file in a text editor and make the necessary changes based on your requirements. For example, you may need to adjust the IP address, password, and service settings. Save the file when you're done.

Step 7: Start the installation: Run the stack.sh script to start the DevStack installation process:

./stack.sh

```
./stack.sh
```

Step 8: Wait for the installation to complete: The installation process may take a while depending on your system's resources and internet speed. DevStack will automatically download and install all the required components for OpenStack.

Step 9: Verify the installation: Once the installation finishes, you can verify if OpenStack is functioning correctly. The stack.sh script should display the admin credentials and the web dashboard URL. Access the dashboard using a web browser and log in with the provided credentials.

VIVA QUESTIONS

1. What is Devstack ?

2. Differentiate between Openstack and Devstack

3. List the Applications of Devstack

4. Differentiate between Openstack and Packstack

RESULT

Thus, usingthe DevStack Openstack setup is implemented

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	**3**	
PREPARATION	**2**	
PERFORMANCE	**5**	
VIVA – VOCE	**5**	
RECORD	**5**	
TOTAL	**20**	

Ex. No: 7	Installation of Single Node Hadoop Cluster
Date:	

AIM

To find procedure to set up the one node Hadoop cluster.

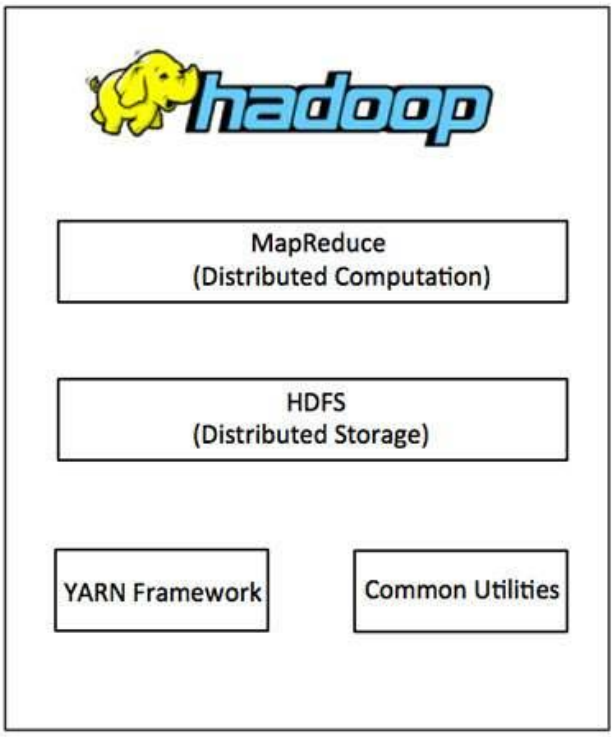
DESCRIPTION

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

Hadoop has two major layers namely

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).



MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte datasets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules −

- **Hadoop Common** − These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** − This is a framework for job scheduling and cluster resource management.

Advantages of Hadoop

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

PROCEDURAL STEPS:

STEP 1: Installing Java is the main prerequisite for Hadoop. Install java1.7

$sudo apt-get update

$sudo apt-get install openjdk-7-jdk

$sudo apt-get install openjdk-7-jre

$ java -version

java version "1.7.0_79"

OpenJDK Runtime Environment (IcedTea 2.5.6) (7u79-2.5.6-0ubuntu1.14.04.1)

OpenJDK 64-Bit Server VM (build 24.79-b02, mixed mode)

STEP 2:

SSH Server accepting password authentication (at least for the setup time)

To install, run:

student@a4cse196:~$ sudo -i

root@a4cse196:~$ passwd root

Enter New Password: Test1234

Re-Enter New Password: Test1234

root@a4cse196:~$ exit

student@a4cse196:~$ su

password: Test1234

root@a4cse196:/home/student# apt-get install openssh-server

STEP 3: Generate the ssh key

root@a4cse196:/home/student# ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa

Generating public/private rsa key pair.

Created directory '/root/.ssh'.

Your identification has been saved in /root/.ssh/id_rsa.

Your public key has been saved in /root/.ssh/id_rsa.pub.

The key fingerprint is:

77:a1:20:bb:db:95:6d:89:ce:44:25:32:b6:81:5d:d5 root@a4cse196

The key's randomart image is:

+--[RSA 2048]----+

| .... |

```
|     o .   E |
|    o B . o  |
|     + * + .  |
|    . S + .  |
|     . o = .  |
|    .  = +   |
|     o = .   |
|    . . o    |
+-----------------+
```

STEP 4:

If the master also acts a slave (`ssh localhost` should work without a password)

root@a4cse196:/home/student# cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

STEP 5: Create Hadoop group and user

5.1 root@a4cse196:/home/student# sudo addgroup hadoop

Adding group `hadoop' (GID 1003) ...

Done.

5.2 root@a4cse196:/home/student# sudo adduser --ingroup hadoop hadoop

Adding user `hadoop' ...

Adding new user `hadoop' (1003) with group `hadoop' ...

Creating home directory `/home/hadoop' ...

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

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

Changing the user information for hadoop

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

root@a4cse196:/home/student#

STEP 6:

Copy **.tar file to home**. (hadoop-2.7.0.tar.gz)

STEP 7: Extracting the tar file

root@a4cse196:/home/student# sudo tar -xzvf hadoop-2.7.0.tar.gz -C /usr/local/lib/

STEP 8: Changing the Ownership

root@a4cse196:/home/student# sudo chown -R hadoop:hadoop /usr/local/lib/hadoop-2.7.0

STEP 9: Create HDFS Directories

root@a4cse196:/home/student# sudo mkdir -p /var/lib/hadoop/hdfs/namenode

root@a4cse196:/home/student# sudo mkdir -p /var/lib/hadoop/hdfs/datanode

root@a4cse196:/home/student# sudo chown -R hadoop /var/lib/hadoop

STEP 10: Check where Java is installed

root@a4cse196:/home/student# readlink -f /usr/bin/java

/usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java

STEP 11: Open "bashrc" and add the code

root@a4cse196:/home/student# gedit ~/.bashrc

Add to ~/.bashrc file:

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

export HADOOP_INSTALL=/usr/local/lib/hadoop-2.7.0

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/native"

STEP 12: Reload "bashrc"

root@a4cse196:/home/student# source ~/.bashrc

STEP 13: Modify JAVA_HOME in /usr/local/lib/hadoop-2.7.0/etc/hadoop/hadoop-env.sh

root@a4cse196:/home/student# cd /usr/local/lib/hadoop-2.7.0/etc/hadoop

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# gedit hadoop-env.sh

export JAVA_HOME=${ JAVA_HOME}

Changed this to below path

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_SSH_OPTS="-p 22"

STEP 14: Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/core-site.xml

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# gedit core-site.xml

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

STEP 15: Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/yarn-site.xml

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# gedit yarn-site.xml

```
<configuration>
  <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> </configuration>

STEP 16: Create /usr/local/lib/hadoop-2.7.0/etc/hadoop/mapred-site.xml from template

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# cp /usr/local/lib/hadoop-2.7.0/etc/hadoop/mapred-site.xml.template /usr/local/lib/hadoop-2.7.0/etc/hadoop/mapred-site.xml

STEP 17: Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/mapred-site.xml

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# gedit mapred-site.xml

```
<configuration>
 <property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
 </property>
</configuration>
```

STEP 18: Modify /usr/local/lib/hadoop-2.7.0/etc/hadoop/hdfs-site.xml

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# gedit hdfs-site.xml

```
<configuration>
 <property>
   <name>dfs.replication</name>
   <value>1</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/var/lib/hadoop/hdfs/namenode</value>
 </property>
 <property>
<name>dfs.datanode.data.dir</name>
<value>file:/var/lib/hadoop/hdfs/datanode</value>
 </property>
</configuration>
```

STEP 19: Make changes in /etc/profile

$gedit /etc/profile

JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

PATH=$PATH:$JAVA_HOME/bin

export JAVA_HOME

export PATH

$source /etc/profile

STEP 20: Format Namenode

root@a4cse196:/usr/local/lib/hadoop-2.7.0/etc/hadoop# hdfs namenode –format

STEP 21: Start SSH, DFS and YARN

sudo service ssh start

start-dfs.sh

yes

yes

start-yarn.sh

root@a4cse196:/home/hadoop# jps

6334 SecondaryNameNode

6498 ResourceManager

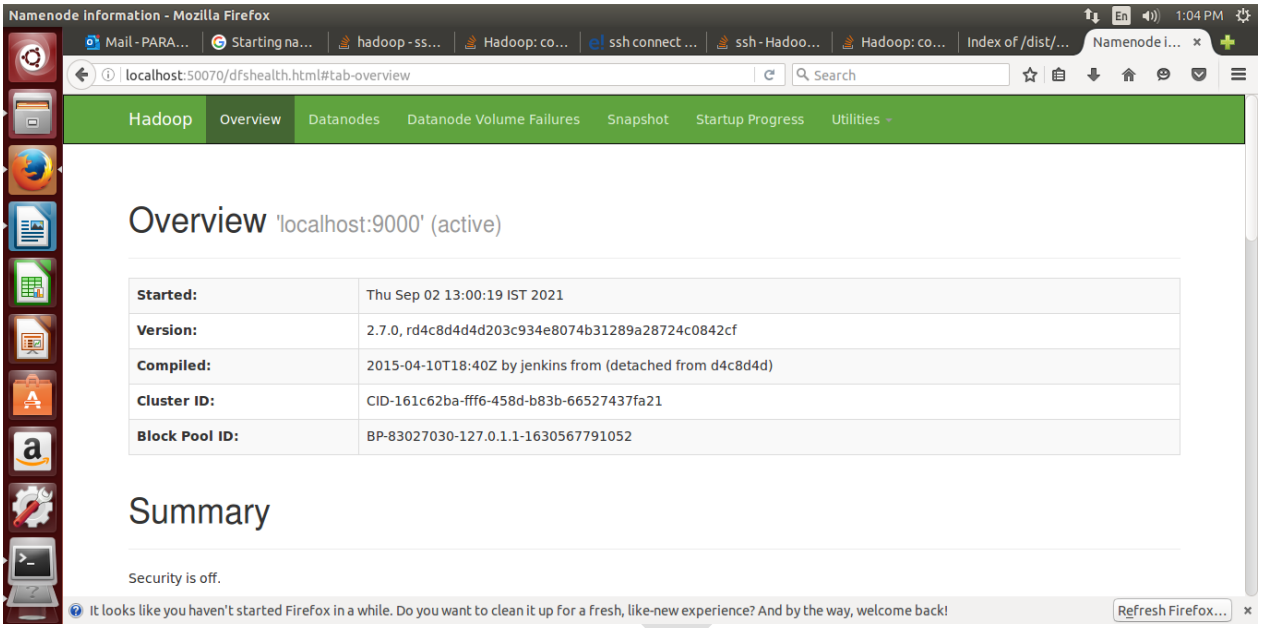
6927 Jps

6142 DataNode

5990 NameNode

6696 NodeManager

STEP 22: Browse the web interface for the Name Node; by default it is available at: http://localhost:50070



VIVA QUESTIONS

1. What is Hadoop?

2. State the types of data.

3. Name three components of Hadoop.

4. In Hadoop, _____ is used for processing / computing the task.

5. In Hadoop, _____ provides a distributed file system that is designed to run on commodity hardware.

RESULT

Thus, the procedure to set up the one node Hadoop cluster was successfully done and verified.

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	**3**	
PREPARATION	**2**	
PERFORMANCE	**5**	
VIVA – VOCE	**5**	
RECORD	**5**	
TOTAL	**20**	

Ex. No: 8	Word Count Program using MapReduce
Date:	

AIM

To count the number of words using JAVA for demonstrating the use of Map and Reduce tasks.

DESCRIPTION

MapReduce is a programming model for writing applications that can process Big Data in parallel on multiple nodes. MapReduce provides analytical capabilities for analysing huge volumes of complex data.

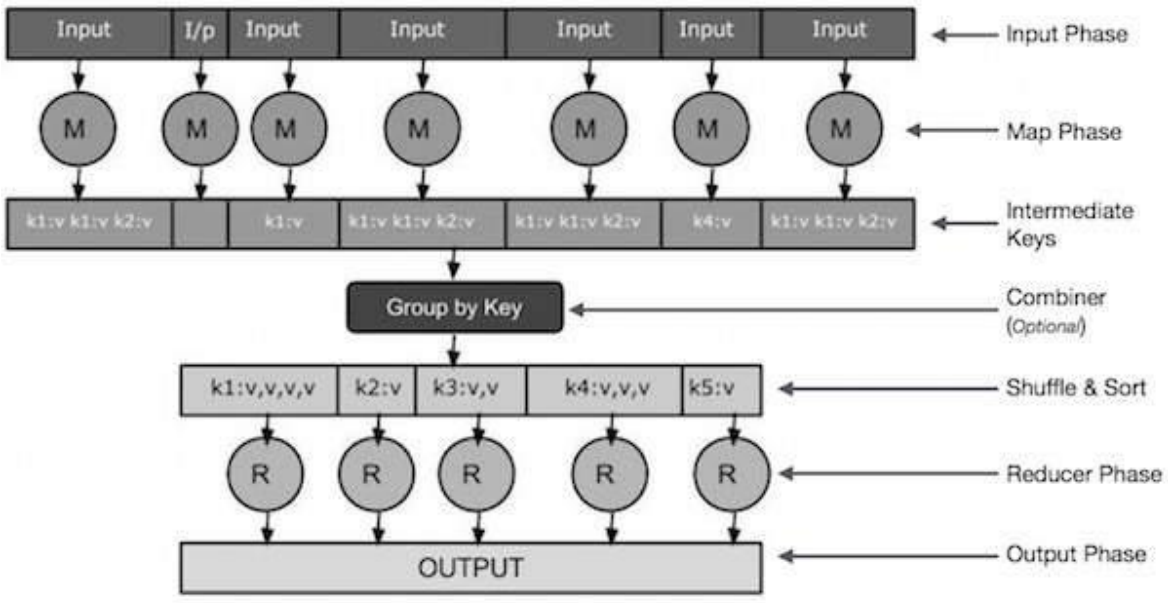
How MapReduce Works?

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduce task is always performed after the map job.

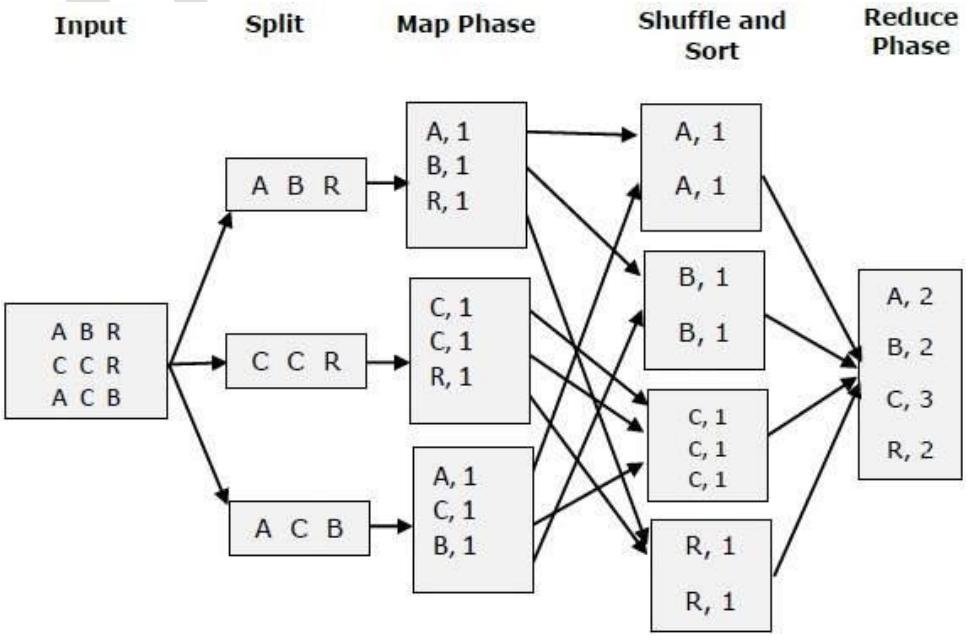
Let us now take a close look at each of the phases and try to understand their significance.



- **Input Phase:** A Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.

- **Map:** Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.
- **Intermediate Keys:** They key-value pairs generated by the mapper are known as intermediate keys.
- **Combiner:** A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.
- **Shuffle and Sort:** The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- **Reducer:** The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.
- **Output Phase:** In the output phase, an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

Example



PROCEDURAL STEPS:

- Analyse the input file content
- Develop the code
    - Writing a map function
    - Writing a reduce function
    - Writing the Driver class
- Compiling the source
- Building the JAR file
- Starting the DFS
- Creating Input path in HDFS and moving the data into Input path
- Executing the program

PROGRAM

WordCount.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
```

```
  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}

public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}

public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
```

```
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Save the program as WordCount.java

STEP 1: Compile the java program

hadoop-core-1.2.1.jar file needed to compile the mapreduce program

https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1

Assuming both jar and java files in same directory run the following command to compile

root@a4cseh160:/#javac -classpath hadoop-core-1.2.1.jar WordCount.java

STEP 2: Create a jar file

Syntax: jar cf jarfilename.jar MainClassName*.class

root@a4cseh160:/#jar cf wc.jar WordCount*.class

STEP 3: Make directory in hadoop file system

Syntax: hdfs dfs -mkdir directoryname

root@a4cseh160:/# hdfs dfs -mkdir /user

STEP 4: Copy the input file into hdfs

Syntax: hdfs dfs -put sourcefile destpath

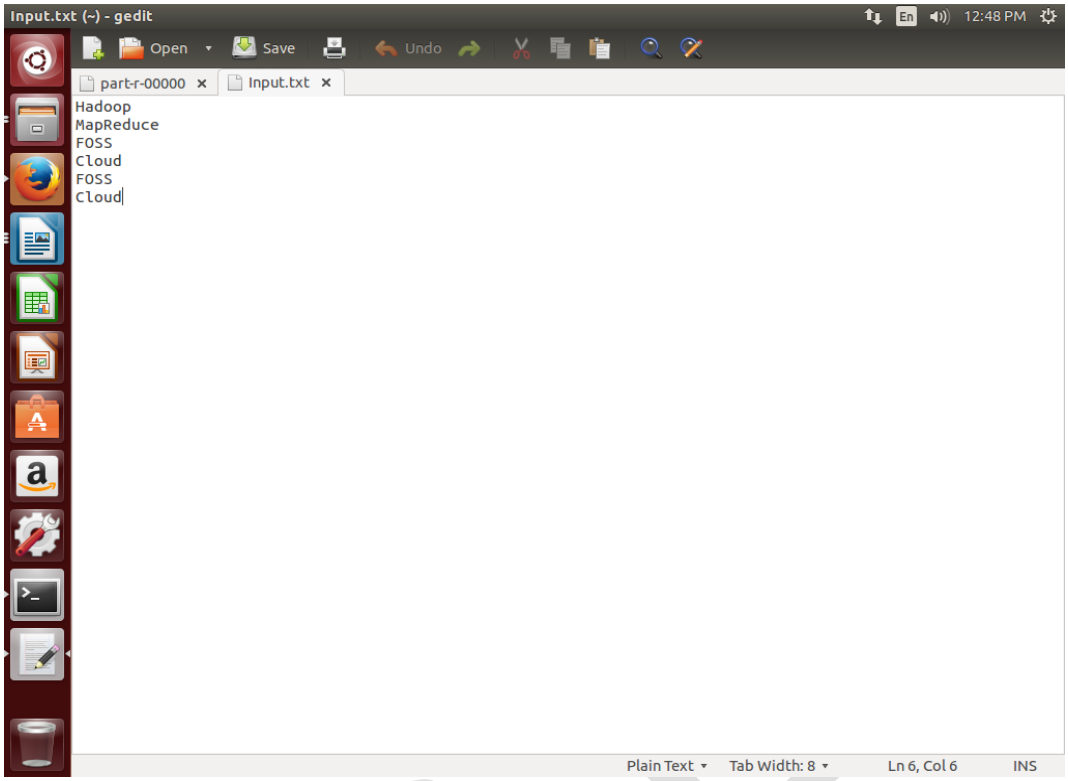
root@a4cseh160:/#hdfs dfs -put /Input.txt /user

STEP 5: To a run a program

Syntax: hadoop jar jarfilename main_class_name inputfile outputpath

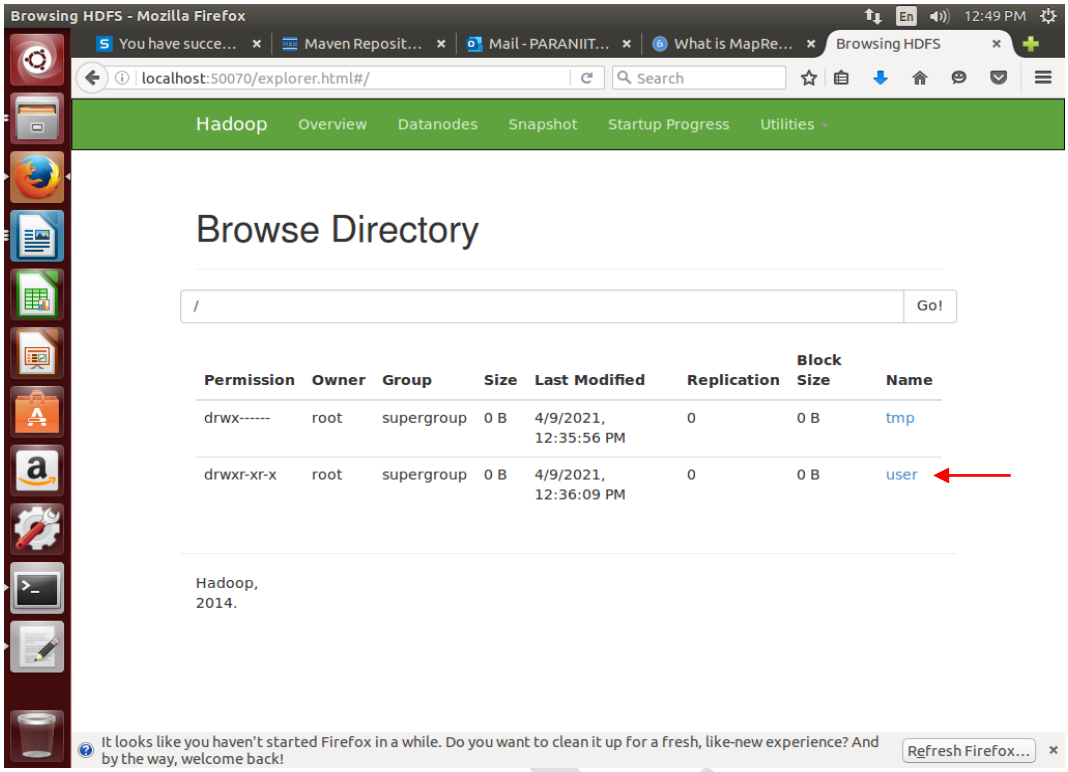
root@a4cseh160:/#hadoop jar wc.jar WordCount /user/Input.txt /user/out

Input File: (Input.txt)

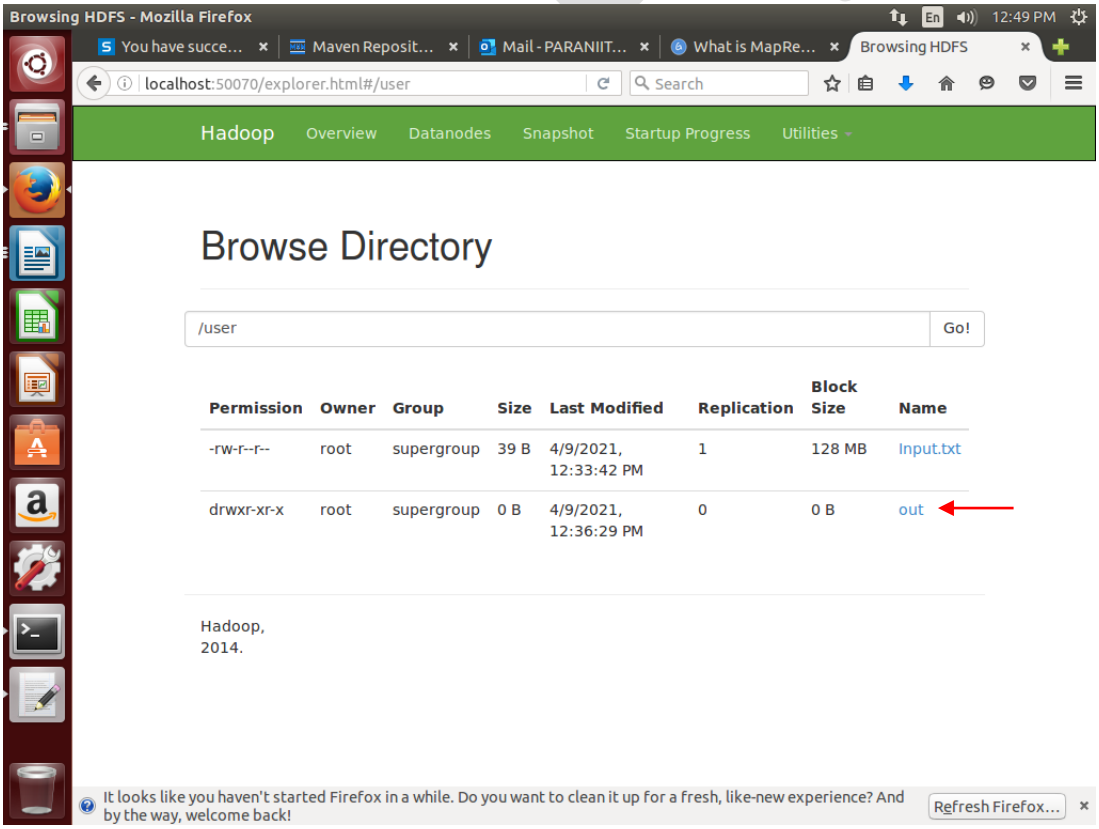


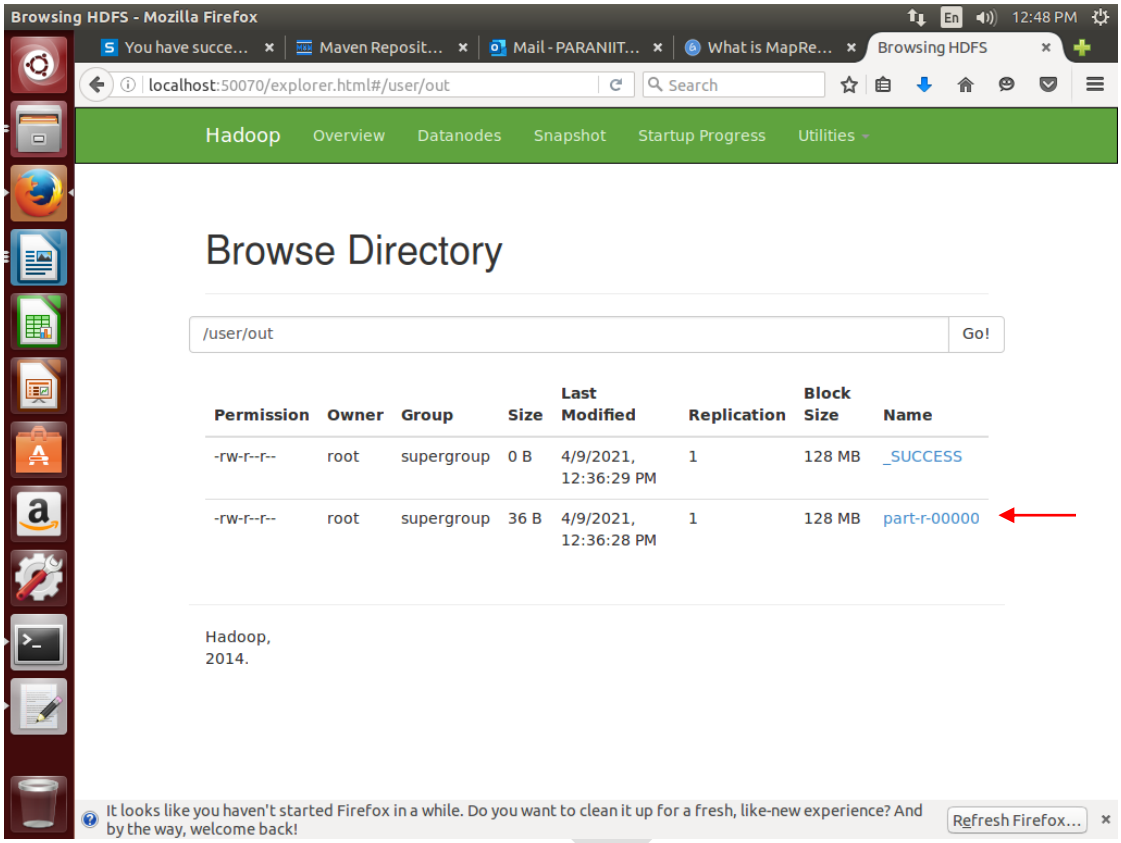
STEP 6: Check the output in the Web UI at http://localhost:50070. In the Utilities tab select browse file system and select the correct user.

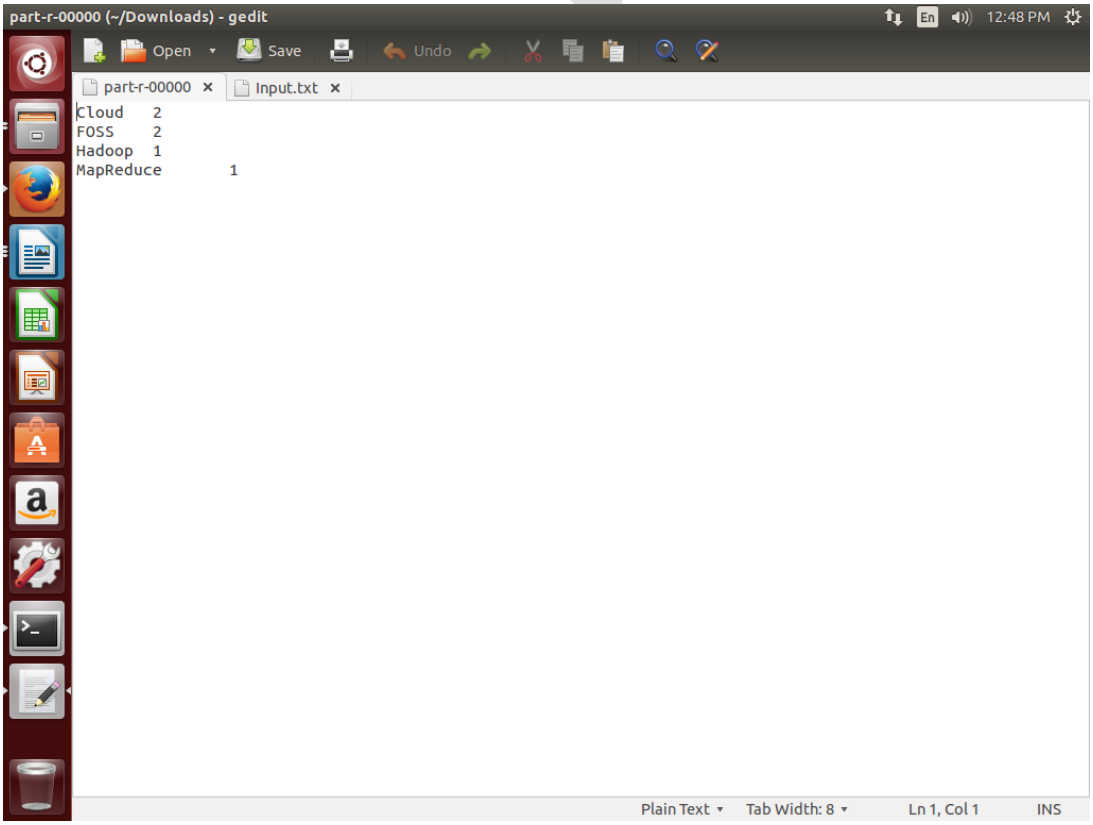
The output is available inside the output folder named **"user"**



Output File: (out)







STEP 7: To Delete a output folder

Syntax: hdfs dfs -rm -R outputpath

root@a4cseh160:/#hdfs dfs -rm -R /user/out.txt

VIVA QUESTIONS

1. What is Hadoop MapReduce?

2. What are the operations performed in MapReduce for computation process?

3. State the benefits of MapReduce.

4. What is Map Task and Reduce Task?

RESULT

Thus, the numbers of words were counted successfully by the use of Map and Reduce tasks.

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	**3**	
PREPARATION	**2**	
PERFORMANCE	**5**	
VIVA – VOCE	**5**	
RECORD	**5**	
TOTAL	**20**	

Ex. No: 9	Implement VM Scheduling Using CloudSim
Date:	

AIM

To simulate VM Scheduling using CloudSim.

DESCRIPTION

CloudSim

- A Framework for modeling and simulation of Cloud Computing Infrastructures and services
- Originally built at the Cloud Computing Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia
- It is completely written in JAVA

Main Features of CloudSim

- Modeling and simulation
- Data center network topologies and message-passing applications
- Dynamic insertion of simulation elements
- Stop and resume of simulation
- Policies for allocation of hosts and virtual machines

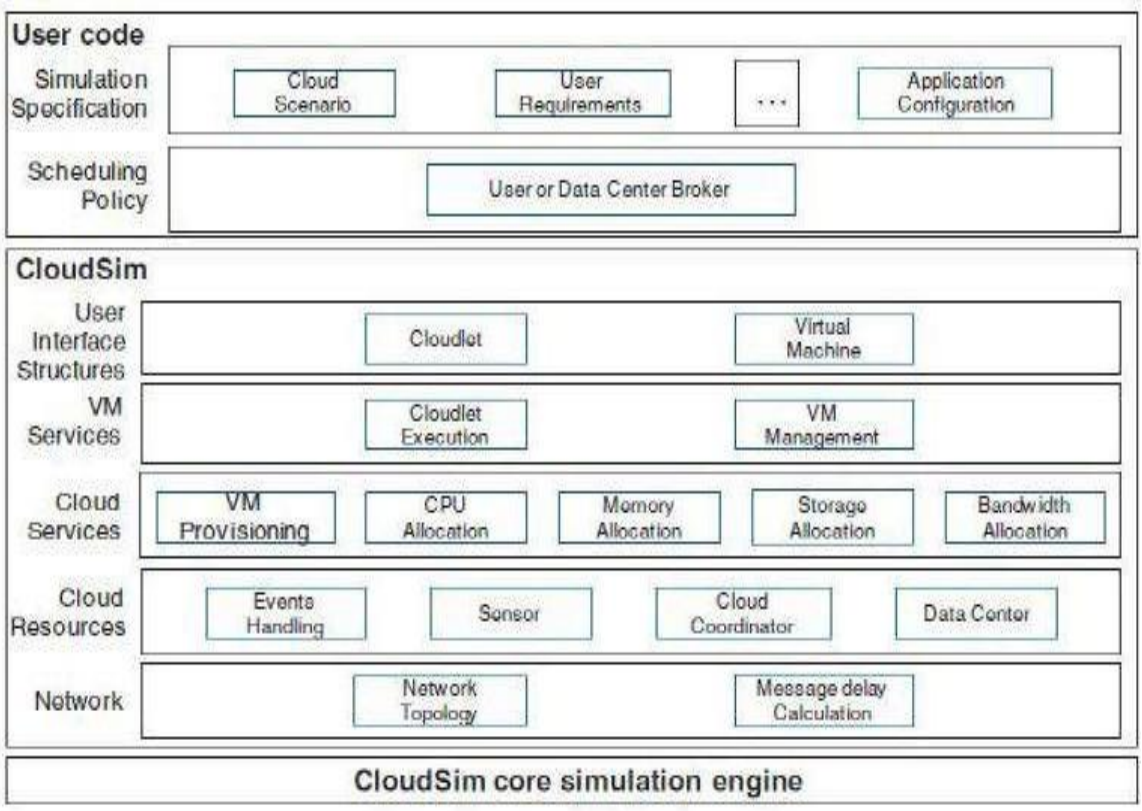
CloudSim – Essentials

- JDK 1.6 or above http://tinyurl.com/JNU-JAVA
- Eclipse 4.2 or above http://tinyurl.com/JNU-Eclipse
- Alternatively NetBeans https://netbeans.org/downloads
- Up & Running with cloudsim guide: https://goo.gl/TPL7Zh

CloudSim – Directory structure

- cloudsim/ -- top level CloudSim directory
- docs/ -- CloudSim API Documentation
- examples/ -- CloudSim examples
- jars/ -- CloudSim jar archives
- sources/ -- CloudSim source code

CloudSim - Layered Architecture



- **Core Functionalities** are queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation clock.
- **CloudSim simulation layer** provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth.
- A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider's defined QoS levels.
- Top-most layer in the CloudSim stack is the User Code that exposes **basic entities for hosts** (number of machines, their specification, and so on), **applications** (number of tasks and their requirements), **VMs**, **number of users and their application types, and broker scheduling policies**.

Datacenter Entity

- The infrastructure-level services (IaaS) related to the clouds can be simulated by **extending the data center entity** of CloudSim.
- Data center entity **manages** several **host entities.**
- Hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider.
- VM policy stands for the operations control policies related to VM life cycle such as provisioning of a host to a VM, VM creation, VM destruction, and VM migration.

Host Entity

- Represents a physical computing server in a Cloud.
- It assigns a pre-configured processing capability (expressed in millions of instructions per second—MIPS), memory, storage, and a provisioning policy for allocating processing cores to VMs.
- The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes.

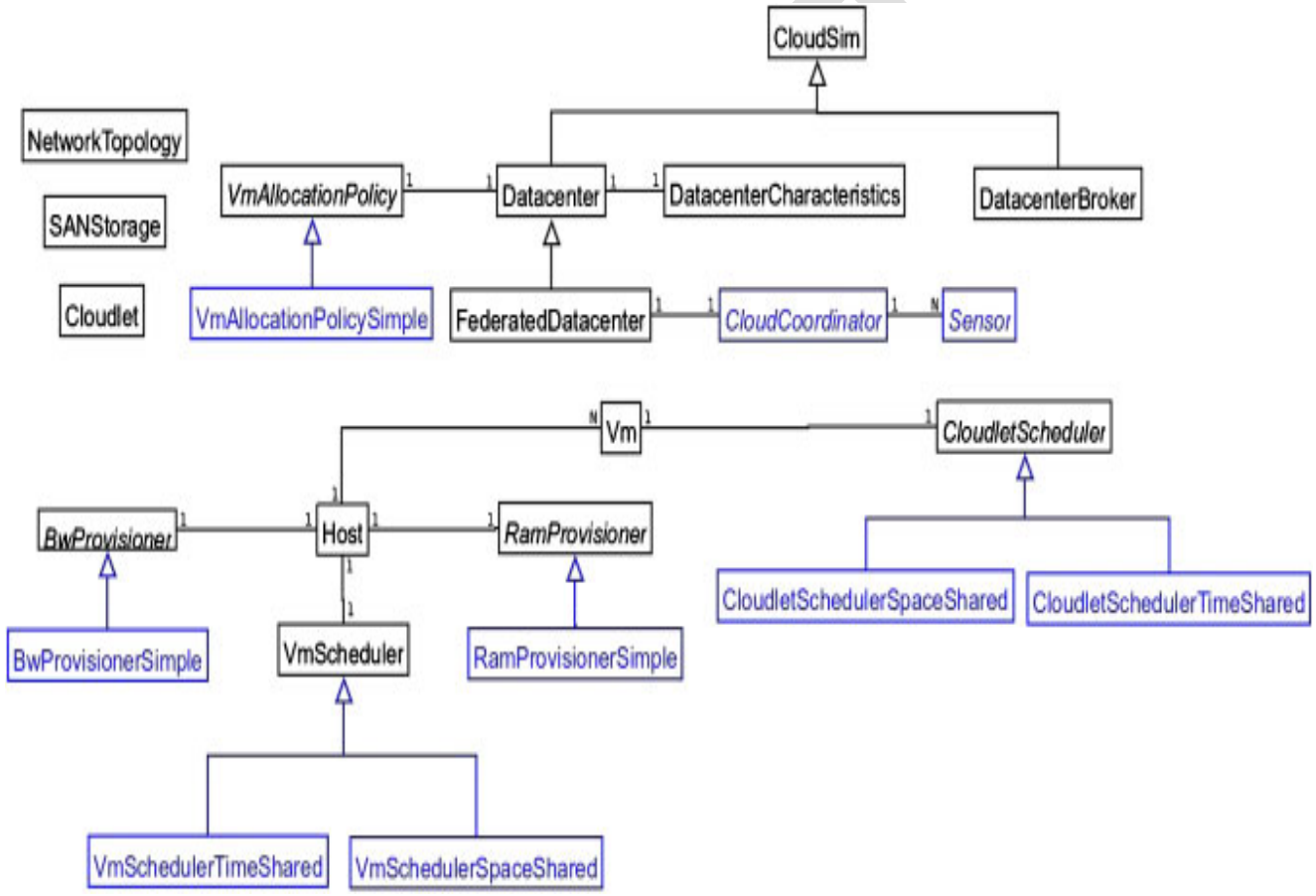
VM Allocation

- Process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider.
- Once an application service (Cloudlet) is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service-specific allocation policy.
- VmAllocationPolicy - VM Allocation controller component to allocate application-specific VMs to hosts in a data center. new policies can be created by the user by extending VmAllocationPolicy Entity.
- Default VmAllocationPolicy is FCFS.
- Hardware requirements, such as the number of processing cores, memory, and storage, form the basis for such provisioning.
- Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in CloudSim.
- Policies used by public Cloud providers (Amazon EC2, Microsoft Azure) are not publicly available

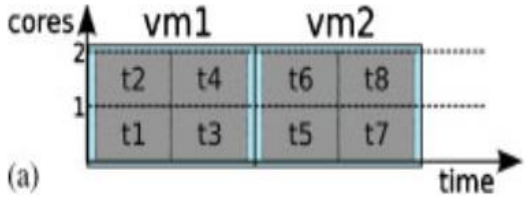
VM scheduler

- Allocation of processing cores to VMs is done by VM Scheduler using host allocation policy.
- Hardware characteristics, such as number of CPU cores, CPU share, and amount of memory (physical and secondary), that are allocated to a given VM instance
- Types:
- **Space Shared:** Assign specific CPU cores to specific VMs
- **Time Shared:** Dynamically distribute the capacity of a core among VMs

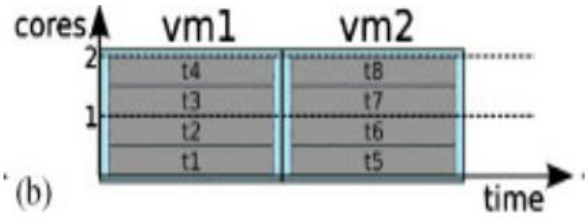
CloudSim – Component Model Classes



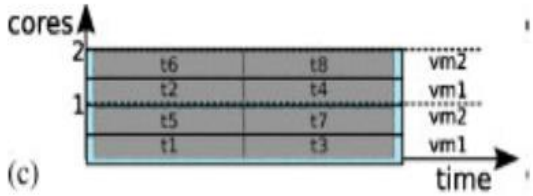
Modelling VM Allocation



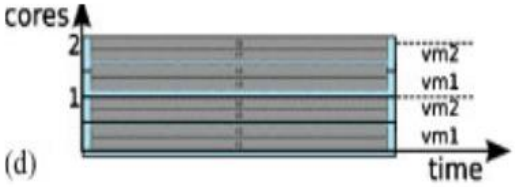
Space-shared provisioning for VMs and tasks



Space-shared provisioning for VMs and time-shared provisioning for tasks

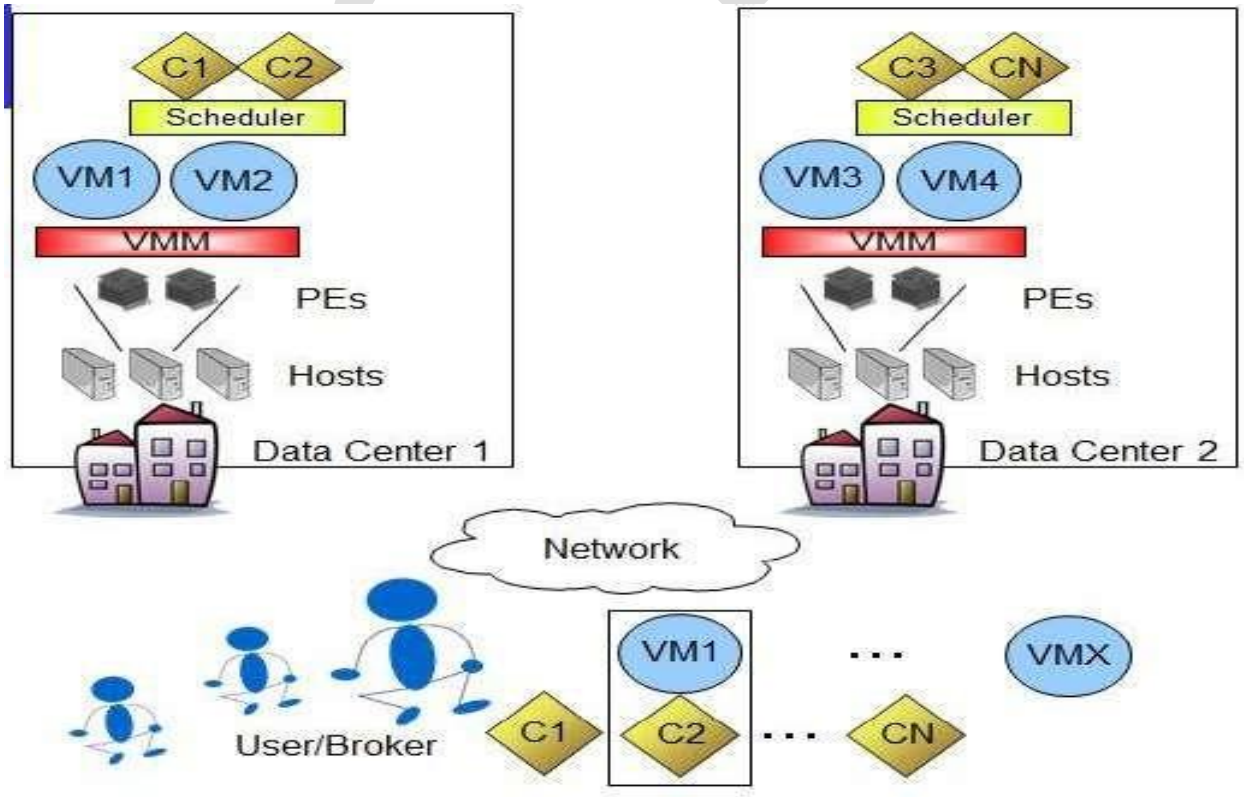


Time-shared provisioning for VMs, space-shared provisioning for tasks



Time-shared provisioning for VMs and tasks

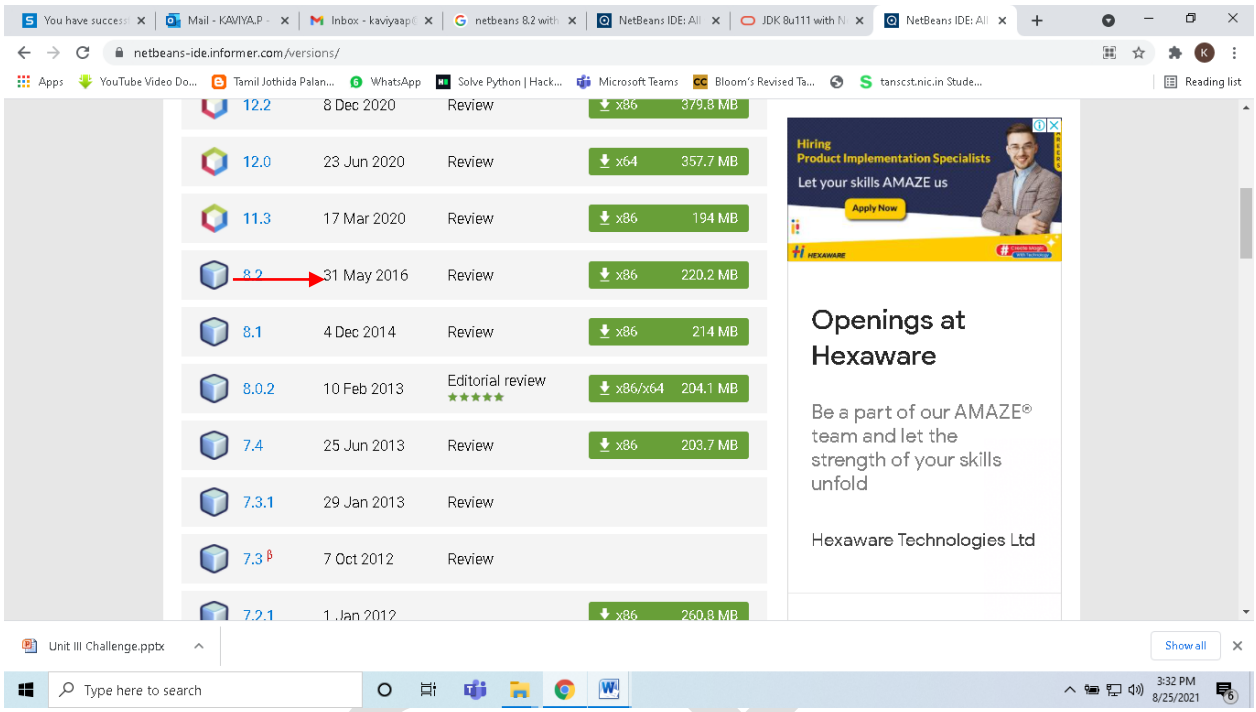
CloudSim Elements / Components



PROCEDURAL STEPS

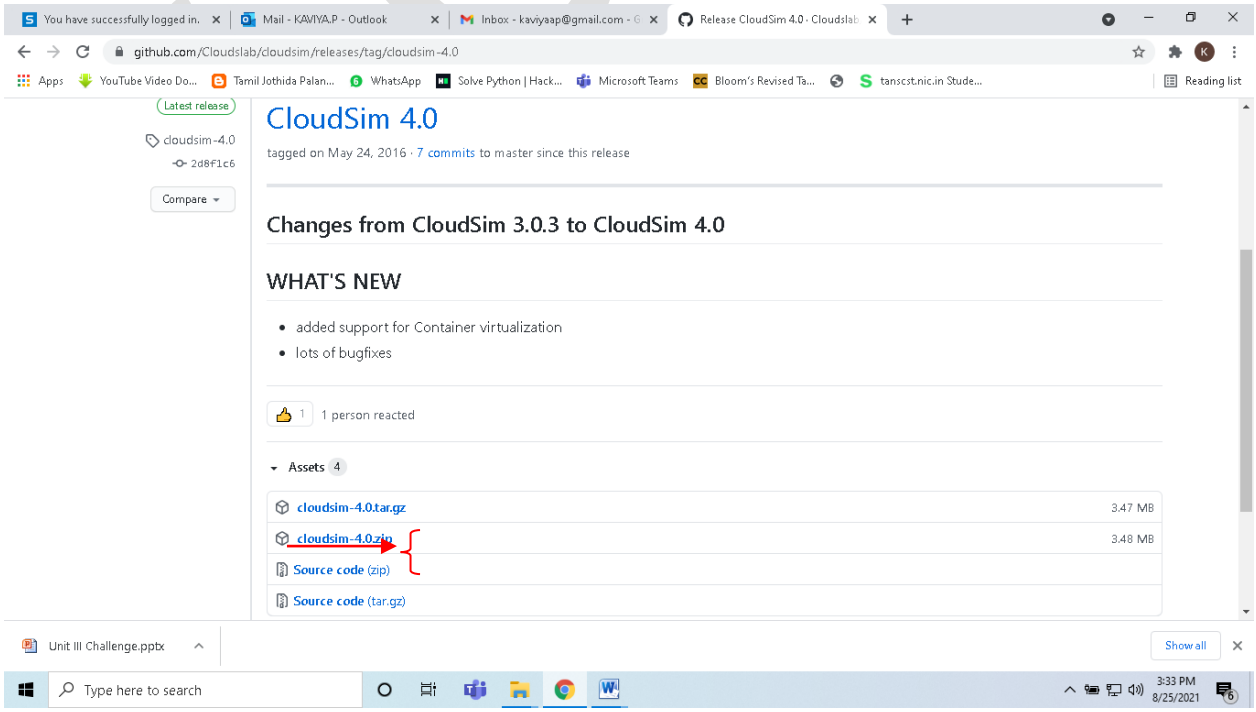
1. Download NetBeans and intall it.

Link: https://netbeans-ide.informer.com/versions/

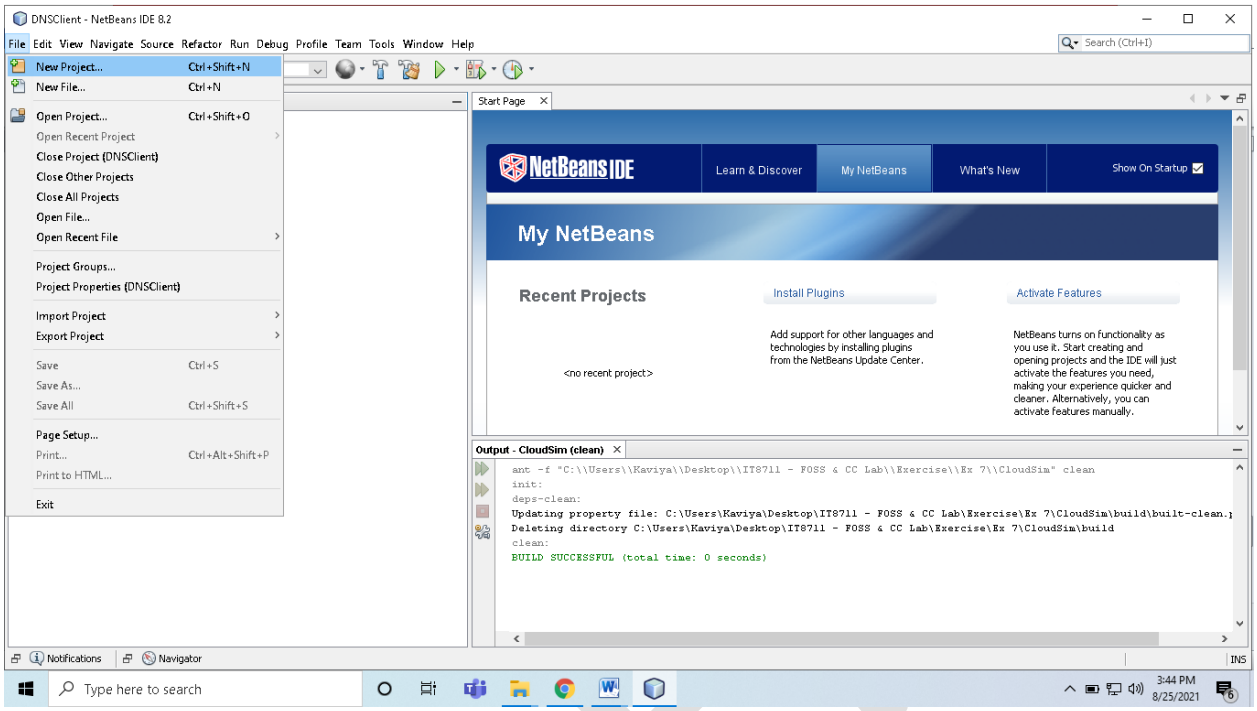


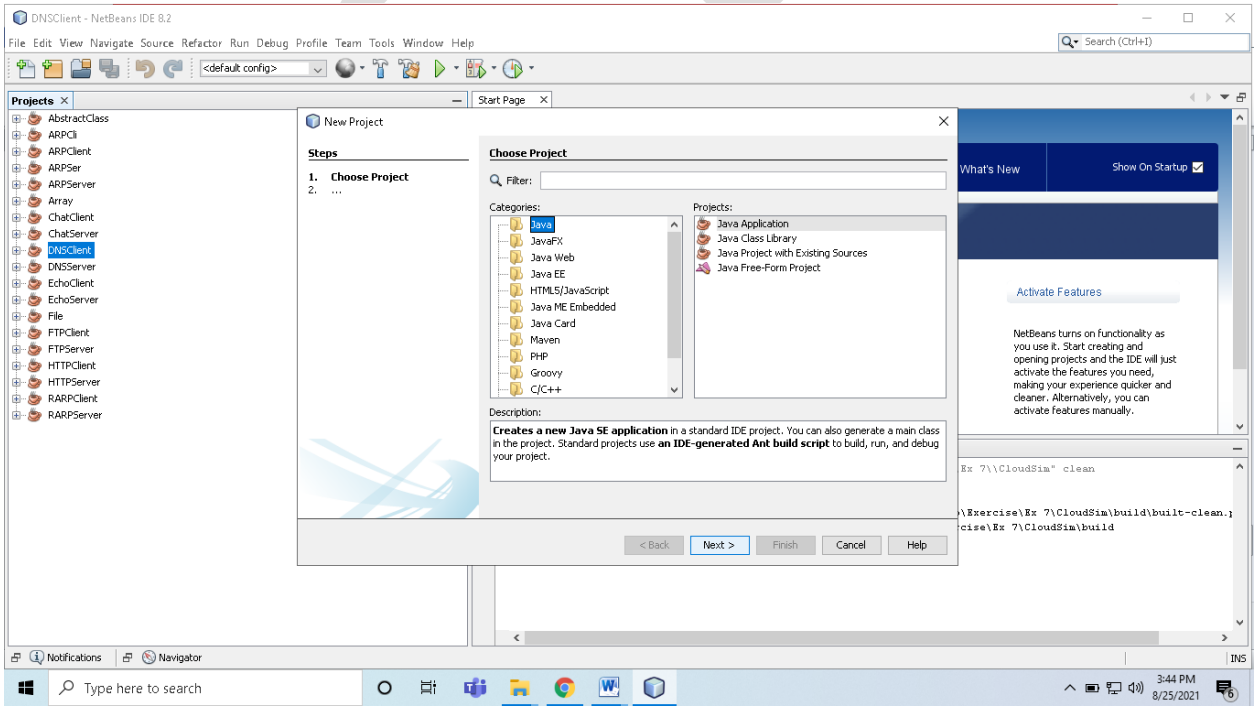
2. Download CloudSim 4.0 (jar and source code), extract and place it in respective folders.

Link: https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-4.0

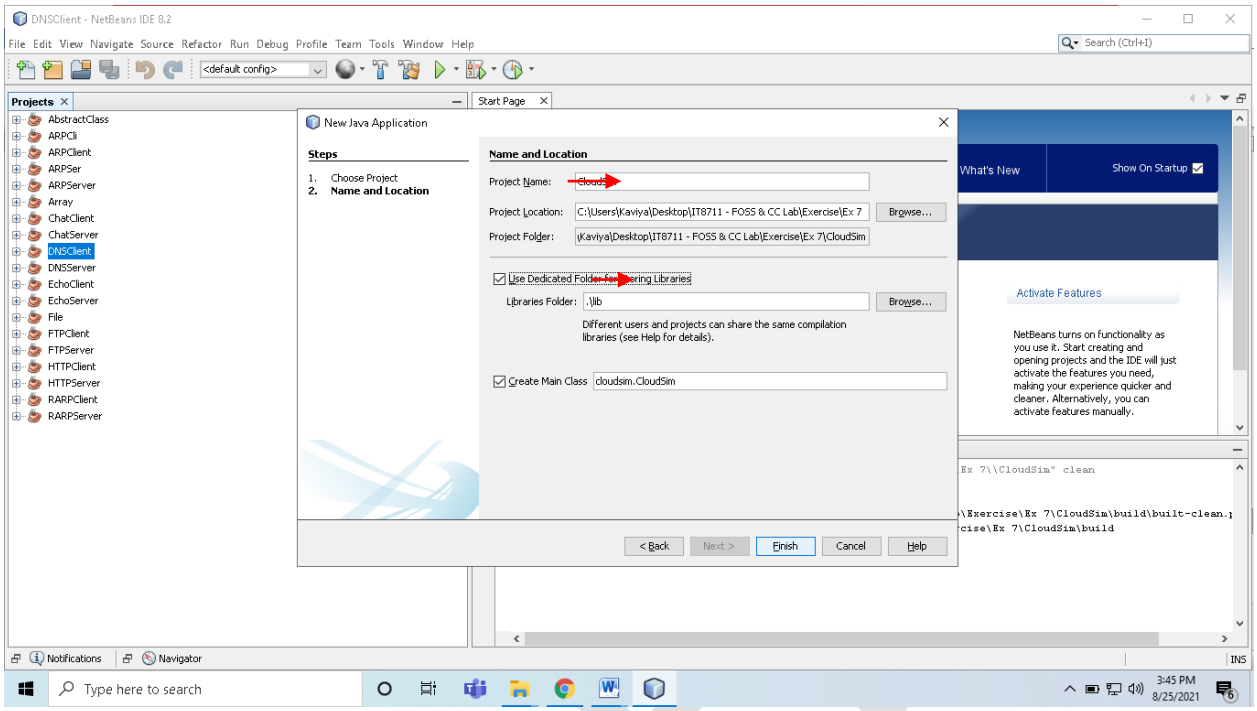


3. Open NetBeans & Create a project “CloudSim”

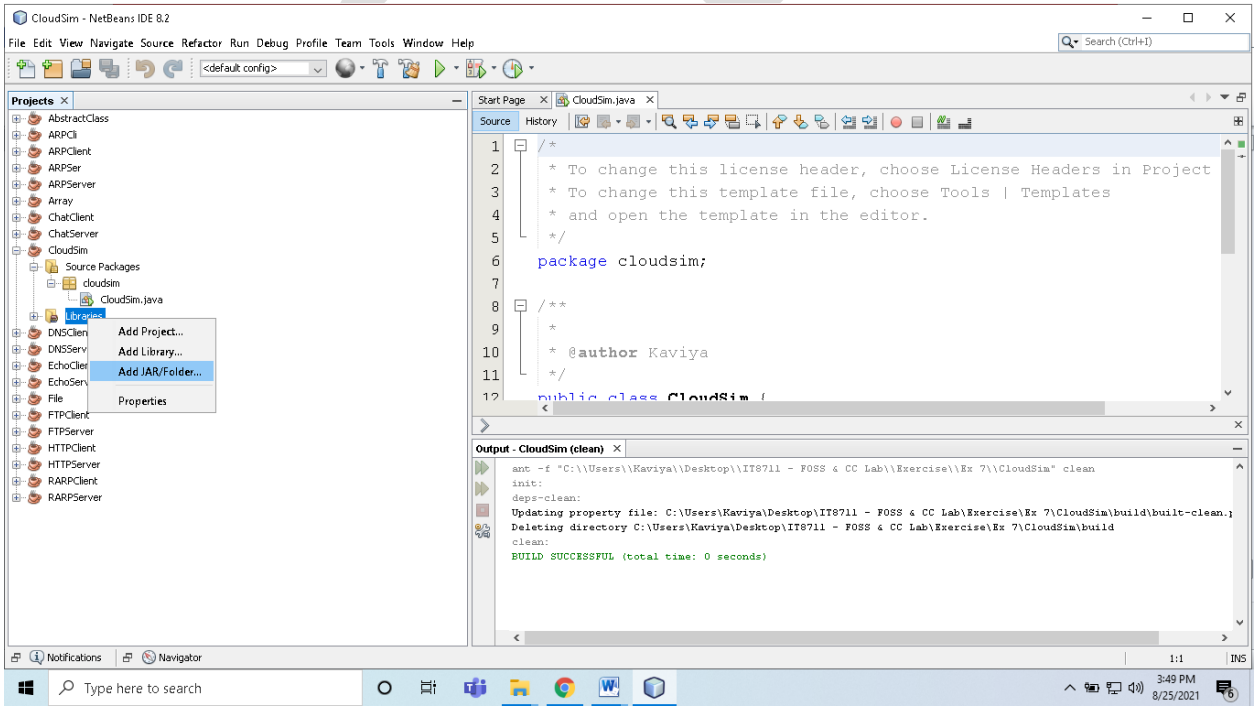


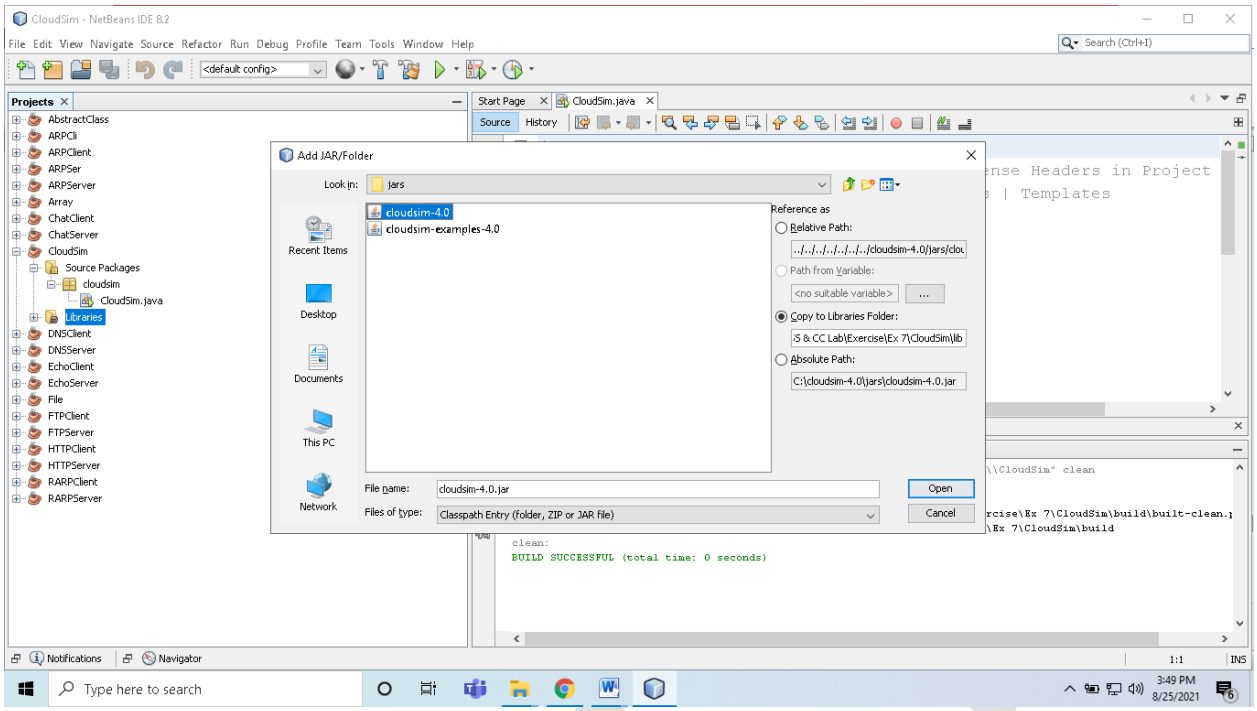


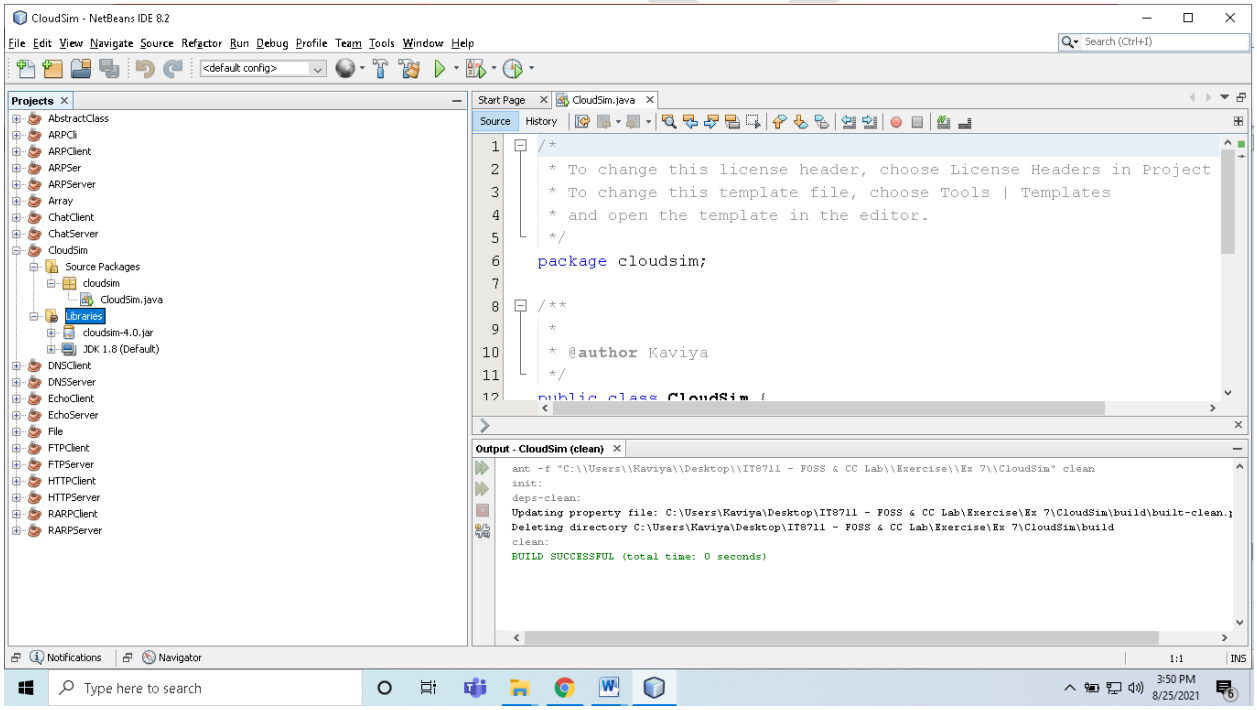
Check “Use dedicated folder for Storing Libraries” → Click “Finish”



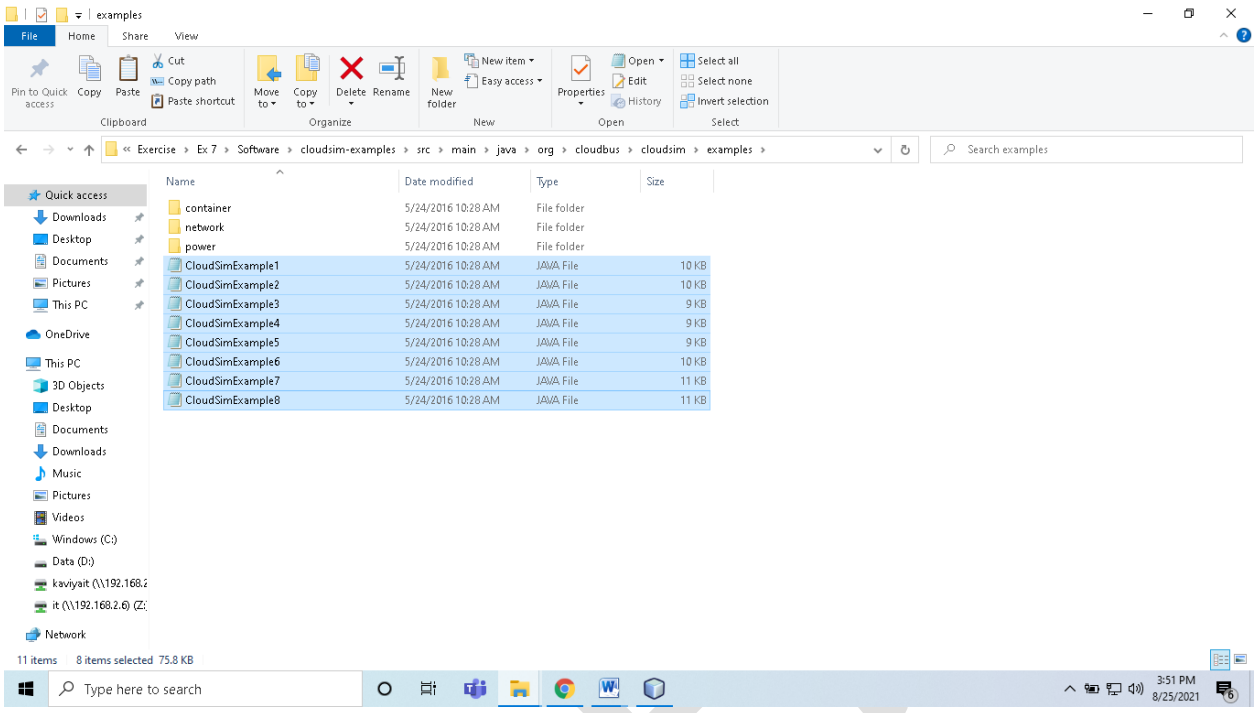
4. Add jar file “cloudsim-4.0”

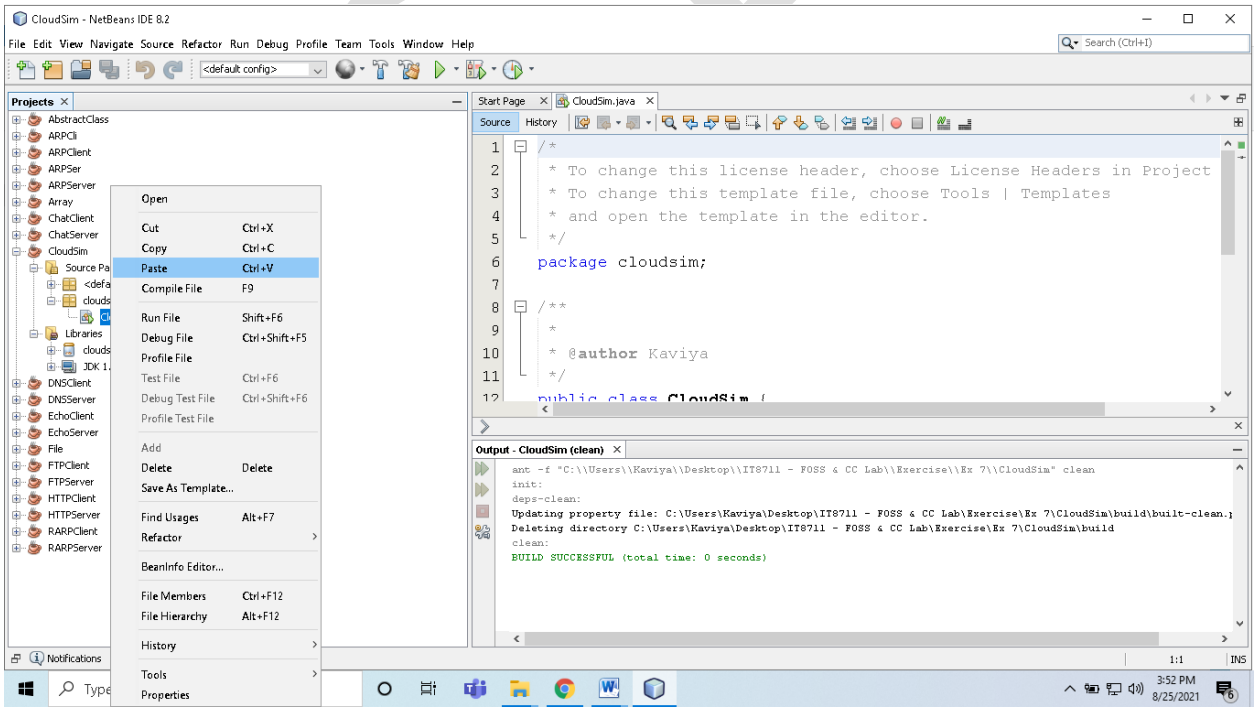


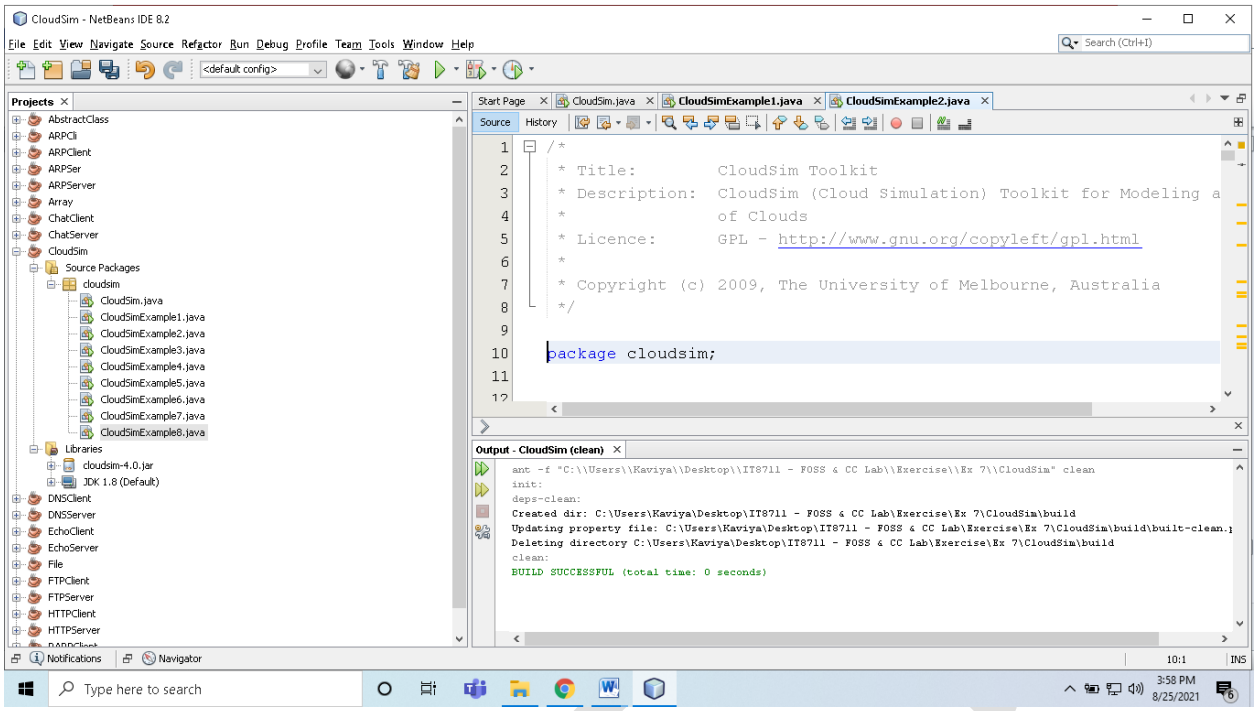

CloudSim - NetBeans IDE 8.2
Add JAR/Folder
Look in:
jars
cloudsim-4.0
cloudsim-examples-4.0
Recent Items
Desktop
Documents
This PC
Network
File name:
cloudsim-4.0.jar
Files of type:
Classpath Entry (folder, ZIP or JAR file)
Open
Cancel
Reference as
Relative Path:
Path from Variable:
Copy to Libraries Folder:
Absolute Path:
C:\cloudsim-4.0\jars\cloudsim-4.0.jar
BUILD SUCCESSFUL (total time: 0 seconds)

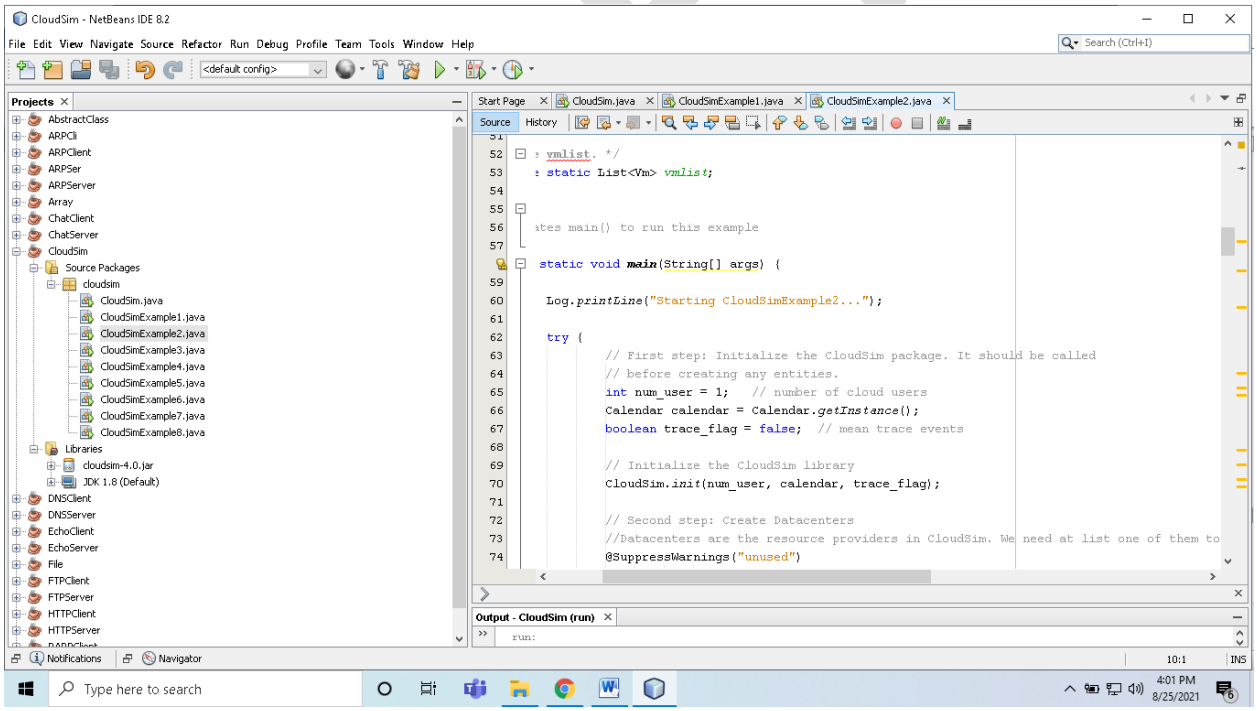

CloudSim - NetBeans IDE 8.2
package cloudsim;
* @author Kaviya
Output - CloudSim (clean)
Updating property file: C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build\built-clean.
Deleting directory C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build
BUILD SUCCESSFUL (total time: 0 seconds)

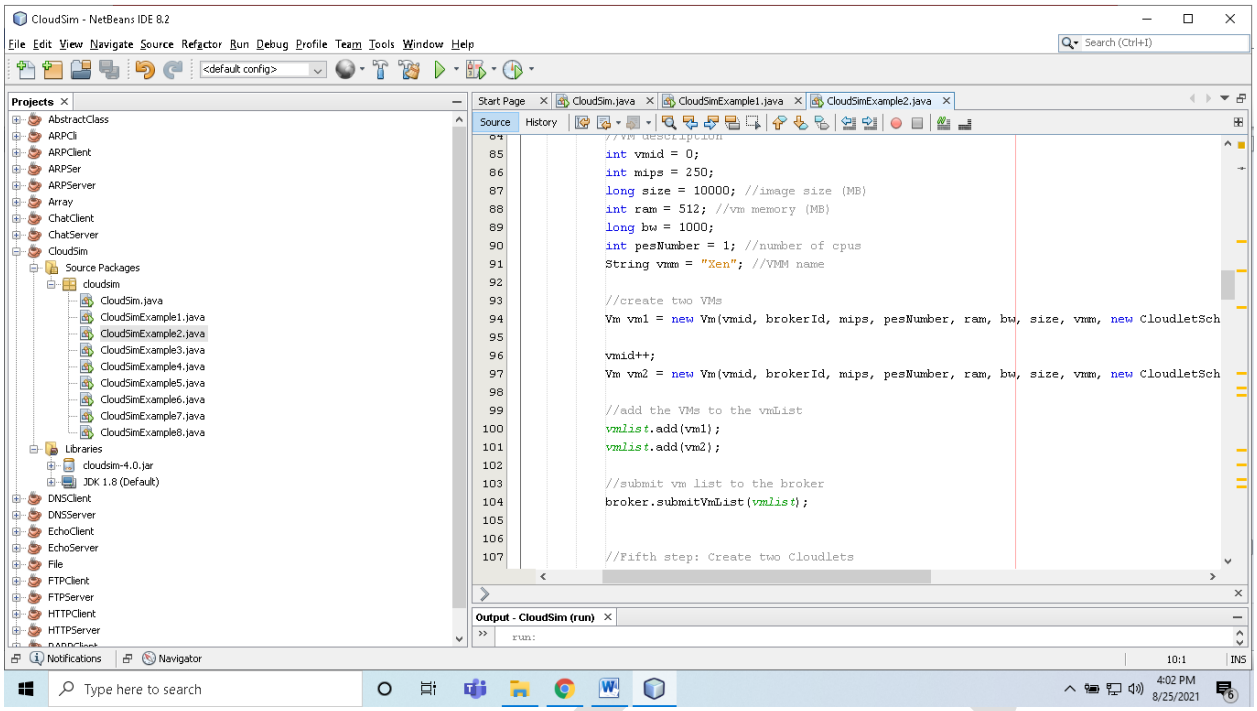
5. Add “cloudsim examples” in CloudSim Project

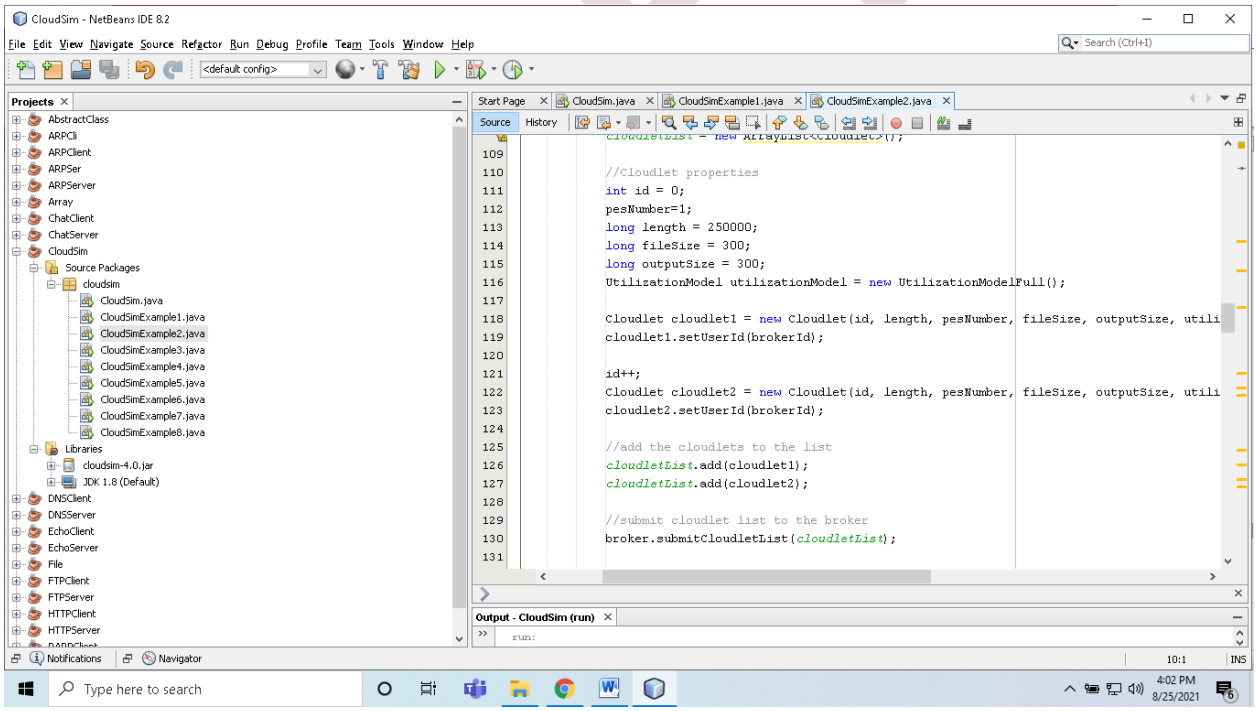


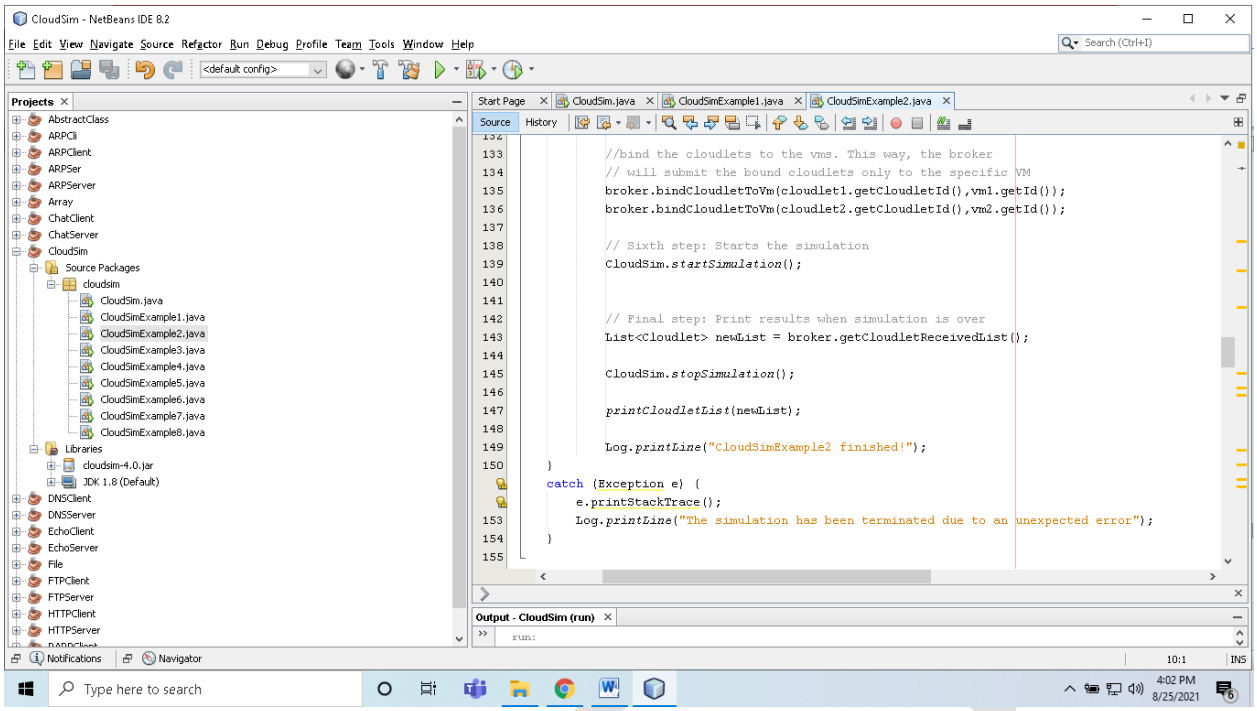



CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
AbstractClass
ARPCli
ARPClient
ARPSer
ARPServer
Array
ChatClient
ChatServer
CloudSim
Source Packages
cloudsim
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
Libraries
cloudsim-4.0.jar
JDK 1.8 (Default)
DNSClient
DNSServer
EchoClient
EchoServer
File
FTPClient
FTPServer
HTTPClient
HTTPServer
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
1 /*
2 * Title: CloudSim Toolkit
3 * Description: CloudSim (Cloud Simulation) Toolkit for Modeling a
4 * of Clouds
5 * Licence: GPL - http://www.gnu.org/copyleft/gpl.html
6 *
7 * Copyright (c) 2009, The University of Melbourne, Australia
8 */
9
10 package cloudsim;
11
Output - CloudSim (clean)
ant -f "C:\\Users\\Kaviya\\Desktop\\IT8711 - FOSS & CC Lab\\Exercise\\Ex 7\\CloudSim" clean
init:
deps-clean:
Created dir: C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build
Updating property file: C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build\built-clean.
Deleting directory C:\Users\Kaviya\Desktop\IT8711 - FOSS & CC Lab\Exercise\Ex 7\CloudSim\build
clean:
BUILD SUCCESSFUL (total time: 0 seconds)
Notifications
Navigator
10:1
INS
Type here to search
3:58 PM
8/25/2021

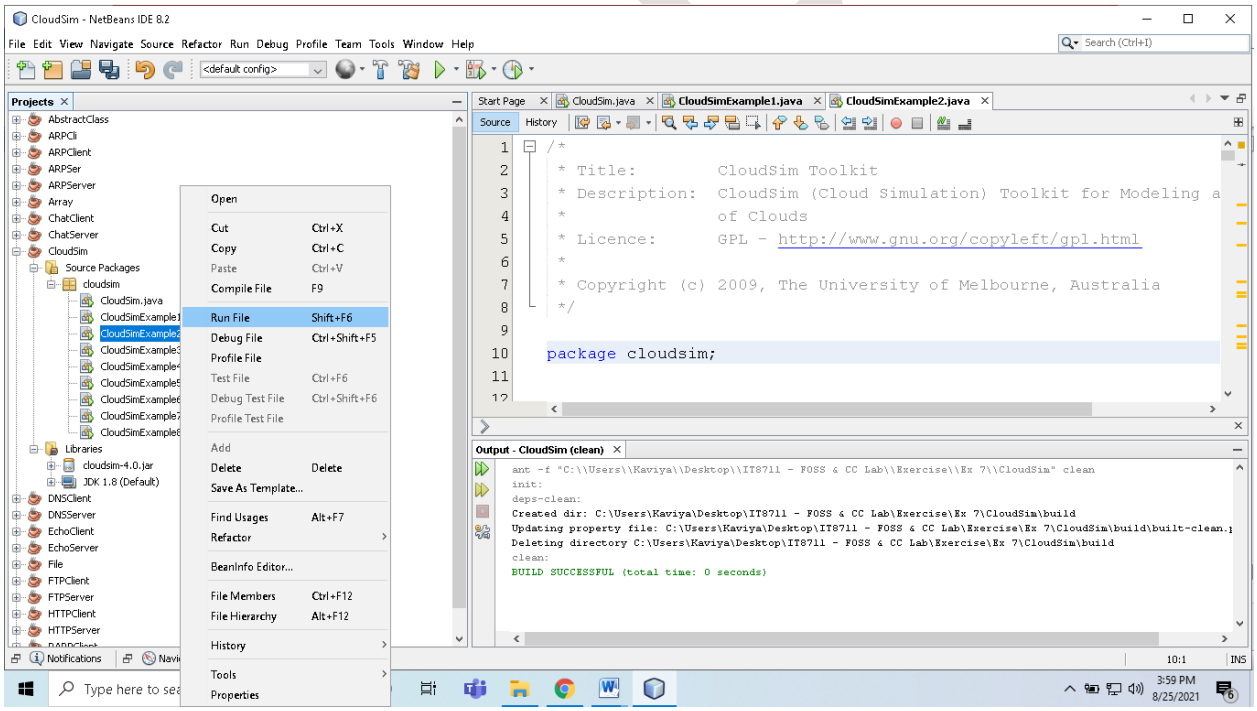

CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
52 * vmlist. */
53 static List<Vm> vmlist;
54
55
56 ates main() to run this example
57
59 static void main(String[] args) {
60
61 Log.printLine("Starting CloudSimExample2...");
62
63 try {
64 // First step: Initialize the CloudSim package. It should be called
65 // before creating any entities.
66 int num_user = 1; // number of cloud users
67 Calendar calendar = Calendar.getInstance();
68 boolean trace_flag = false; // mean trace events
69
70 // Initialize the CloudSim library
71 CloudSim.init(num_user, calendar, trace_flag);
72
73 // Second step: Create Datacenters
74 //Datacenters are the resource providers in CloudSim. We need at list one of them to
@SuppressWarnings("unused")
Output - CloudSim (run)
run:
Notifications
Navigator
10:1
INS
Type here to search
4:01 PM
8/25/2021


CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
AbstractClass
ARPCli
ARPClient
ARPSer
ARPServer
Array
ChatClient
ChatServer
CloudSim
Source Packages
cloudsim
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
Libraries
cloudsim-4.0.jar
JDK 1.8 (Default)
DNSClient
DNSServer
EchoClient
EchoServer
File
FTPClient
FTPServer
HTTPClient
HTTPServer
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
//VM description
int vmid = 0;
int mips = 250;
long size = 10000; //image size (MB)
int ram = 512; //vm memory (MB)
long bw = 1000;
int pesNumber = 1; //number of cpus
String vmm = "Xen"; //VMM name
//create two VMs
Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSch
vmid++;
Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSch
//add the VMs to the vmList
vmlist.add(vm1);
vmlist.add(vm2);
//submit vm list to the broker
broker.submitVmList(vmlist);
//Fifth step: Create two Cloudlets
Output - CloudSim (run)
run:
Notifications
Navigator
10:1
INS
Type here to search
4:02 PM
8/25/2021

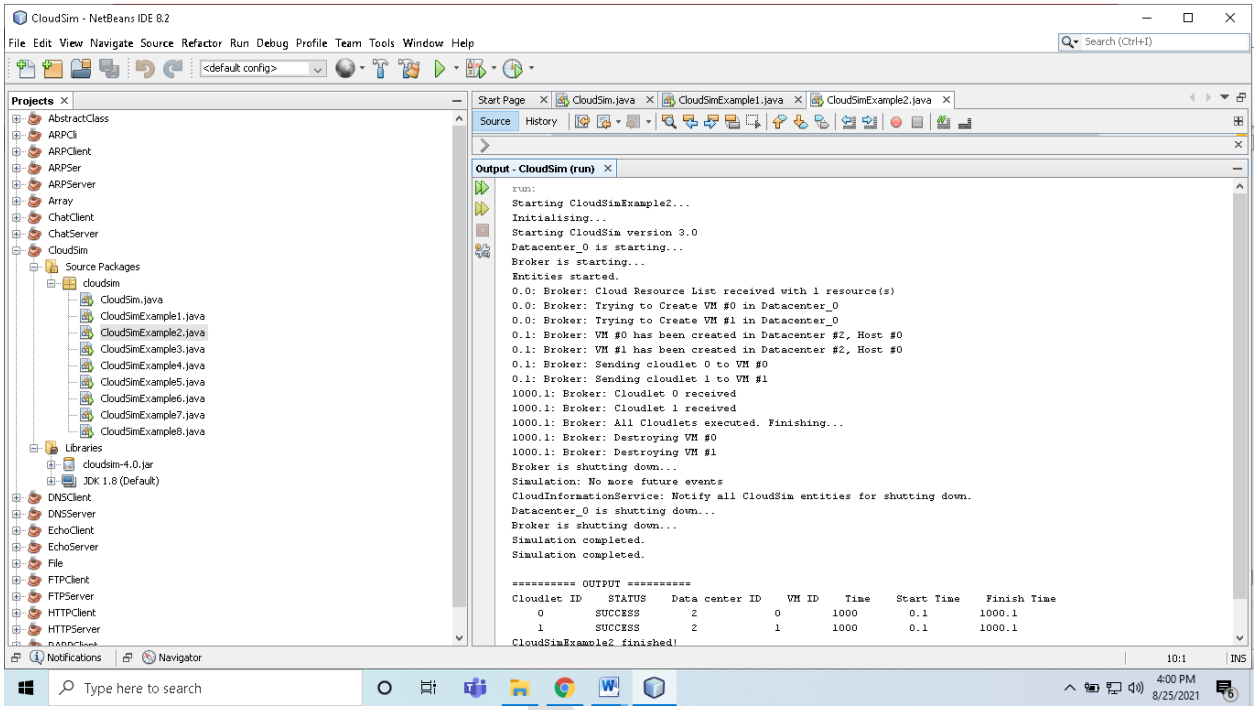

CloudSim - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
AbstractClass
ARPCli
ARPClient
ARPSer
ARPServer
Array
ChatClient
ChatServer
CloudSim
Source Packages
cloudsim
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
Libraries
cloudsim-4.0.jar
JDK 1.8 (Default)
DNSClient
DNSServer
EchoClient
EchoServer
File
FTPClient
FTPServer
HTTPClient
HTTPServer
Start Page
CloudSim.java
CloudSimExample1.java
CloudSimExample2.java
Source
History
//Cloudlet properties
int id = 0;
pesNumber=1;
long length = 250000;
long fileSize = 300;
long outputSize = 300;
UtilizationModel utilizationModel = new UtilizationModelFull();
Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utili
cloudlet1.setUserId(brokerId);
id++;
Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utili
cloudlet2.setUserId(brokerId);
//add the cloudlets to the list
cloudletList.add(cloudlet1);
cloudletList.add(cloudlet2);
//submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);
Output - CloudSim (run)
run:
Notifications
Navigator
10:1
INS
Type here to search
4:02 PM
8/25/2021



6. Run the CloudSim examples



7. Output for CloudSimExample2.java



VIVA QUESTIONS

1. What is CloudSim?

2. CloudSim is written in _____.

3. _____ class represents any task that is run on a VM.

4. _____ is responsible for functioning of VMs, including VM creation, management, destruction, and submission of cloudlets to the VM.

5. _____ is the default VM allocation policy.

RESULT

Thus, the VM scheduling algorithms in CloudSim are studied and executed.

Observation by students: (what student able to?)

1.

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	3	
PREPARATION	2	
PERFORMANCE	5	
VIVA – VOCE	5	
RECORD	5	
TOTAL	20	

Ex. No: 10	Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories using Git Bash and Git Hub
Date:	

AIM

To simulate VM Scheduling using CloudSim.

PROCEDURAL STEPS:

Git commands list

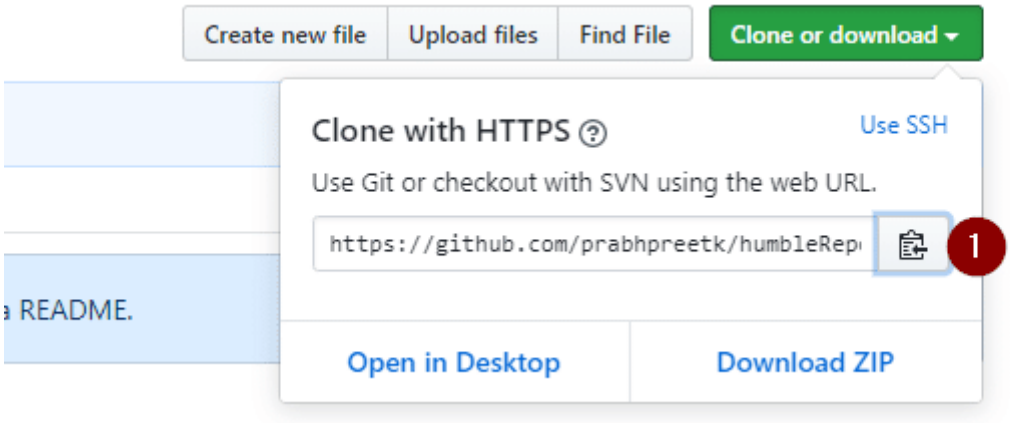
- git clone
- git commit
- git push
- git fetch
- git pull
- git checkout
- git reset

git clone

Usage: git clone [URL]

Suppose, we want to work on a file that is on a remote Github repository as another developer. How can we do that? We can work on this file by clicking on Clone or Download and copying the link and pasting it on the terminal with the git clone command. This will import the files of a project from the remote repository to our local system.

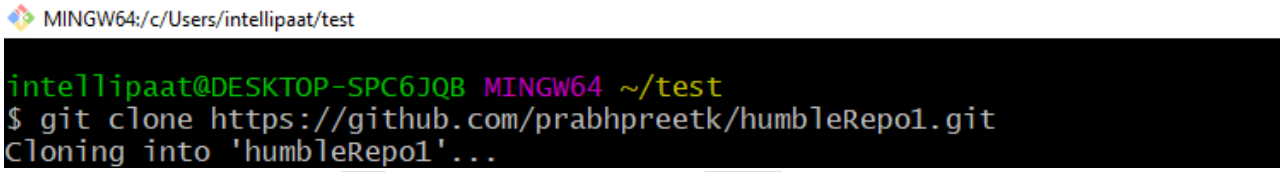
(1) The below screenshot shows from where to copy the link:



To create a local folder, we have to use the following command:

```
mkdir [directory- name]
cd [directory- name]
git clone [URL]
```

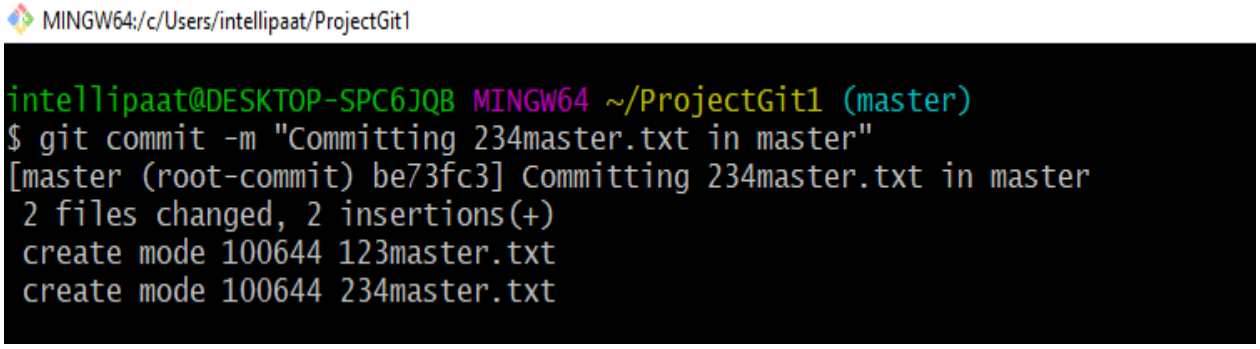
Now, paste the copied link along with the git clone command as shown below:



git commit

Usage: git commit -m “message”

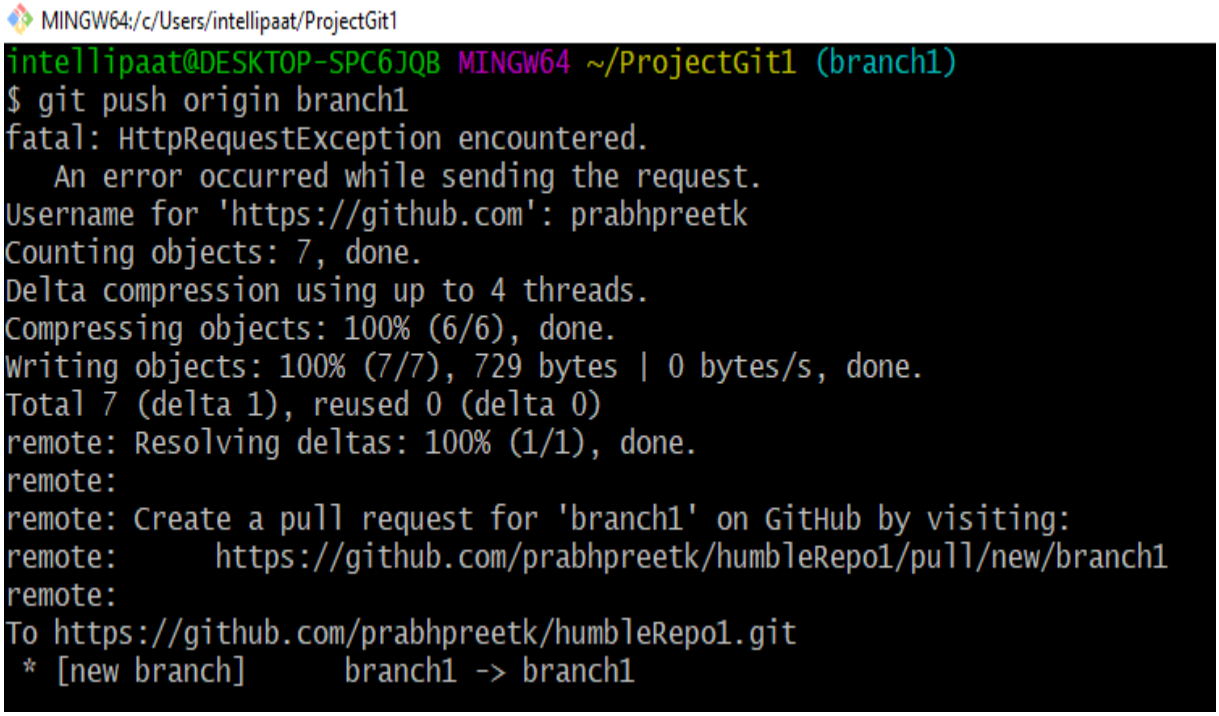
This command records or snapshots files permanently in the version history. All the files, which are there in the directory right now, are being saved in the Git file system.



git push

Usage: git push origin [branch name]

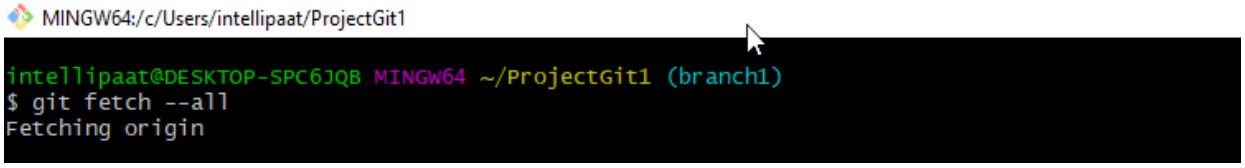
Suppose, we have made some changes in the file and want to push the changes to our remote repository on a particular branch. By using the command 'git push,' the local repository's files can be synced with the remote repository on Github.



git fetch

Usage: git fetch

When we use the command git fetch, Git gathers any commit from the target branch that does not exist in our current branch and stores it in our local repository. However, it does not merge it with our current branch.

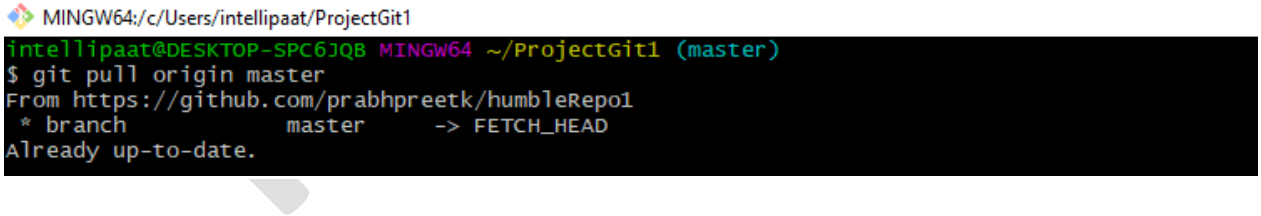


This is particularly useful when we need to keep our repository up to date but are working on something that might break if we updated our files. To integrate the commits into our master branch, we use merge. It fetches all of the branches from the repository. It also downloads all the required commits and files from another repository.

git pull

Usage: git pull origin master

The git pull command first runs ‘git fetch’ which downloads the content from the specified remote repository and then immediately updates the local repo to match the content.



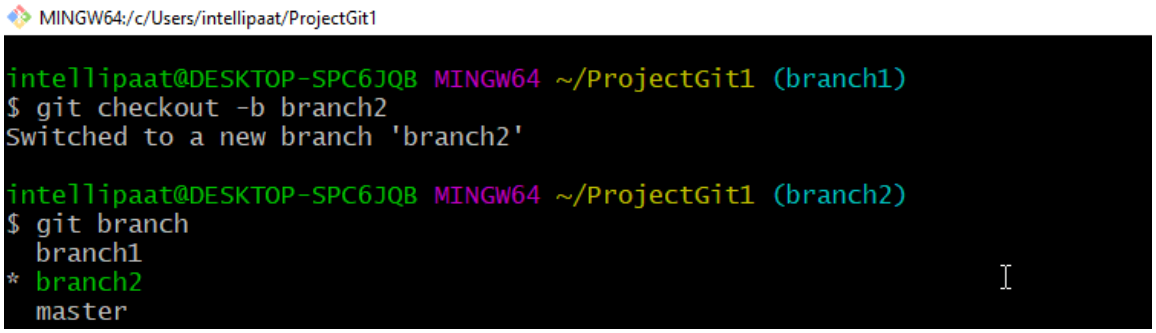
git checkout

Usage (i): git checkout [name-of-the-new-branch]

We use this command to navigate to an existing branch, add new files, and commit the files:

Usage (ii): git checkout -b [name-of-the-new-branch]

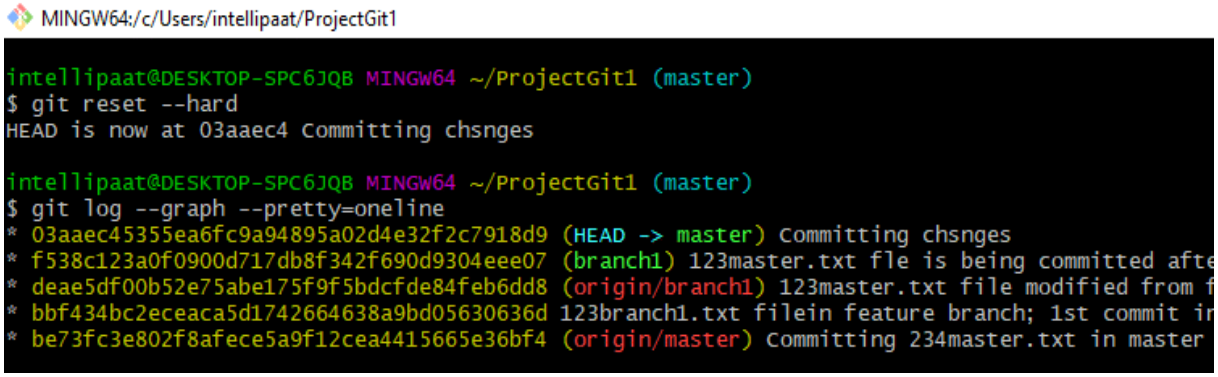
We use this command to create a branch and navigate to that particular branch (say, the 'name-of-the-new-branch' is 'branch2') at the same time:



git reset

Usage: git reset –hard [SOME-COMMIT]

We use this command to return the *entire* working tree to the last committed state.



This will discard commits in a private branch or throw away the uncommitted changes! Here, we have executed a 'hard reset' using the –hard option. Git displays the output indicating that the HEAD is pointing to the latest commit. Now, when we check the state

with git status, Git will indicate that there are no pending changes (if any prior, and the addition of a new file, if not staged, will be destroyed. It is critical to take note that this data loss cannot be undone.

If we do git reset –hard [SOME-COMMIT], then Git will:

- Make our current branch (typically master) back to point <SOME-COMMIT>
- Make the files in our working tree and the index ("staging area") the same as the versions committed at <SOME-COMMIT>

VIVA QUESTIONS

1. What is Github repository?

2. What are the advantages of using Github?

3. What are the types of accounts available in Github?

4. What is the role of Github in project development?

CATEGORY	MAX. MARKS ALLOTED	MARKS AWARDED
DISCIPLINE	**3**	
PREPARATION	**2**	
PERFORMANCE	**5**	
VIVA – VOCE	**5**	
RECORD	**5**	
TOTAL	**20**	