



SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]

[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]

[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]

[ISO 9001-2015 Certified and all eligible programmes Accredited by NBA]

VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

16CS269 – CLOUD COMPUTING LABORATORY

LAB RECORD

ACADEMIC YEAR: 2022-2023

BATCH: 2019-2023

NOVEMBER 2022

SRI RAMAKRISHNA ENGINEERING COLLEGE



[Educational Service: SNR Sons Charitable Trust]
[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001-2015 Certified and all eligible programmes Accredited by NBA]
VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022



Department of Computer Science and Engineering

CLASS: IV B.E. CSE

SEMESTER: VII

Certified that this is the Bonafide record of work done by

Ms. _____ Priyadarshini. R. V _____ in the **16CS269 – CLOUD COMPUTING**

LABORATORY of this institution for VII Semester during the academic year 2022-2023.

Faculty In-Charge

Dr. P Mathiyalagan, AP/CSE

Professor & Head

Dr. A. Grace Selvarani

Date: _____.

REGISTER NUMBER

1901149

Submitted for the VIIth semester B.E. Practical Examination held on _____

Internal Examiner

External Examiner

INDEX

S.NO	DATE	TITLE OF THE EXPERIMENT	PAGE NO	MARKS	FACULTY SIGNATURE
1	11.07.2022	A study on cloud computing environment and virtualization	3		
2	18.07.2022	Find procedure to run the virtual machine of different configuration, to check how many virtual machines can be utilized at a particular time	7		
3	23.07.2022	Find a procedure to attach a virtual block and to the virtual machine and check whether it holds the data even after the release of the virtual machine	11		
4	25.07.2022	Install C compiler in the virtual machine and execute a sample program	14		
5	01.08.2022	Show the virtual machine migration based on the certain condition from one node to the other.	16		
6	08.08.2022	Find procedure to install storage controller and interact with it.	24		
7	22.08.2022	Find procedure to set up the one node Hadoop cluster.	34		
8	05.09.2022	Mount the one node hadoop cluster using fuse	40		
9	19.09.2022	Write a program to use the API's of Hadoop to interact with it	42		

10	08.10.2022	Write a word count program to demonstrate the use of Map and Reduce tasks	46		
11	10.10.2022	Find procedure for file transfer between virtual machines	50		
12	17.10.2022	Hosting a static webpage in google appengine	55		
13	31.10.2022	Open Nebula Installation	61		
		TOTAL			

EX. NO: 01**DATE:11.7.22****A STUDY ON CLOUD COMPUTING ENVIRONMENT AND
VIRTUALIZATION****AIM**

To provide overview of cloud computing, architecture and different types of cloud computing.

CLOUD COMPUTING

Cloud computing is a type of computing that relies on shared computing resources rather than having local servers or personal devices to handle applications. Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.

BENEFITS OF CLOUD COMPUTING

1. Efficiency / cost reduction
2. Data security
3. Scalability
4. Mobility
5. Disaster Recovery
6. Control
7. Competitive Edge
8. Flexibility
9. Sustainability
10. Loss Prevention

CLOUD COMPUTING SERVICES

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)

Infrastructure as a Service:

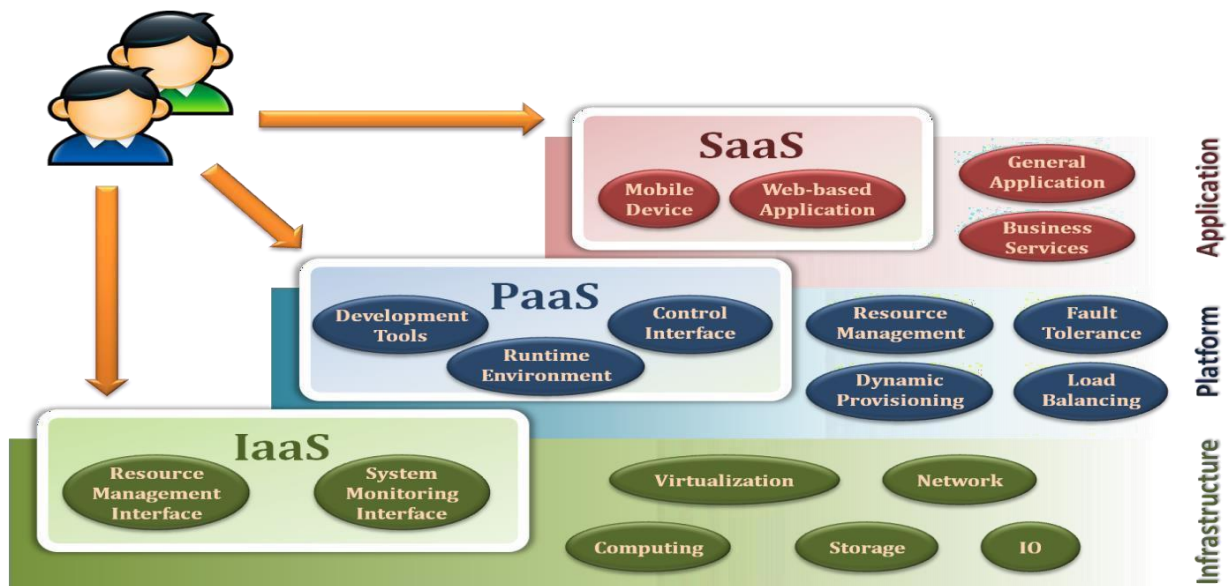
Infrastructure as a Service, or IaaS, gives business access to vital web architecture, such as storage space, servers, and connections, without the business need of purchasing and managing this internet infrastructure themselves. Because of the economies of scale and specialization involved, this can be to the benefit of both the business providing the infrastructure and the one using it. In particular, IaaS allows an internet business a way to develop and grow on demand. Both PaaS and SaaS clouds are grounded in IaaS clouds, as the company providing the software as service is also providing the infrastructure to run the software. Choosing to use an IaaS cloud demands a willingness to put up with complexity, but with that complexity comes flexibility. Amazon EC2 and Rackspace Cloud are examples of IaaS.

Platform as a Service:

Platform as a Service (PaaS) clouds are created, many times inside IaaS Clouds by specialists to render the scalability and deployment of any application trivial and to help make expenses scalable and predictable. Some examples of a PaaS system include: Mosso, Google App Engine, and Force.com. The chief benefit of a service like this is that for as little as no money you can initiate your application with no stress more than basic development and maybe a little porting if we are dealing with an existing app. Furthermore, PaaS allows a lot of scalability by design because it is based on cloud computing as defined earlier in the article. To learn operations staff, a PaaS can be very useful if app will capitulate. The most important negative of using a PaaS Cloud provider is that these services may implement some restrictions or trade-offs that will not work with your product under any circumstances.

Software as a Service:

Software as a Service (SaaS) is relatively mature, and the phrase's use predates that of cloud computing. Cloud applications allow the cloud to be leveraged for software architecture, reducing the burdens of maintenance, support, and operations by having the application run on computers belonging to the vendor. GMail and Salesforce are among examples of SaaS run as clouds, but not all SaaS has to be based in cloud computing.



DEPLOYMENT MODEL

Public cloud:

Public cloud As the name suggests, this type of cloud deployment model supports all users who want to make use of a computing resource, such as hardware (OS, CPU, memory, storage) or software (application server, database) on a subscription basis. Most common uses of public clouds are for application development and testing, non-mission-critical tasks such as file-sharing, and e-mail service.

Private cloud:

Private cloud True to its name, a private cloud is typically infrastructure used by a single organization. Such infrastructure may be managed by the organization itself to support various user groups, or it could be managed by a service provider that takes care of it either on-site or off-site. Private clouds are more expensive than public clouds due to the capital expenditure involved in acquiring and maintaining them. However, private clouds are better able to address the security and privacy concerns of organizations today.

Hybrid cloud:

Hybrid cloud In a hybrid cloud, an organization makes use of interconnected private and public cloud infrastructure. Many organizations make use of this model when they need to scale up their IT infrastructure rapidly, such as when leveraging public clouds to supplement the capacity available within a private cloud. For example, if an online retailer needs more

computing resources to run its Web applications during the holiday season it may attain those resources via public clouds.

Community cloud:

Community cloud This deployment model supports multiple organizations sharing computing resources that are part of a community; examples include universities cooperating in certain areas of research, or police departments within a county or state sharing computing resources. Access to a community cloud environment is typically restricted to the members of the community.

LIMITATIONS

- In-transit Security
- Communication Infrastructure Redundancy
- Knowing the Lines of Responsibility
- Downtime
- Security and Privacy
- Vulnerability to attack
- Limited control and flexibility
- Vendor lock-in

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT

Thus, the overview on cloud computing types. Services, limitations has been studied.

EX. NO: 02**DATE:18.7.22**

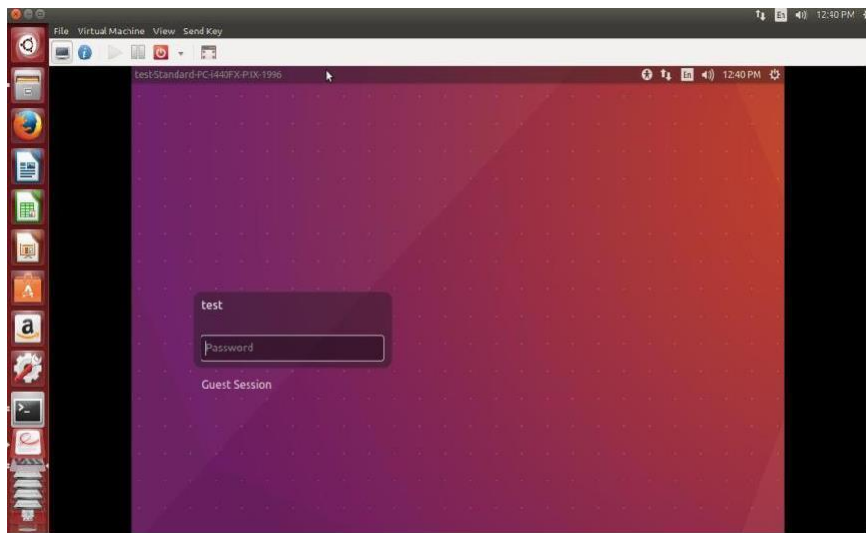
FIND PROCEDURE TO RUN THE VIRTUAL MACHINE OF DIFFERENT CONFIGURATION, TO CHECK HOW MANY VIRTUAL MACHINES CAN BE UTILIZED AT PARTICULAR TIME

AIM:

To understand procedure to run the virtual machine of different configuration. Check how many Virtual machines can be utilized at particular time.

PROCEDURE:**OPENSTACK**

OpenStack is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service, whereby virtual servers and other resources are made available to customers.

To Login in Guest OS**TO RUN VM IN OPEN STACK**

Step 1 : Under the Project Tab, **Click Instances**. In the right side screen Click Launch Instance.

Step 2 : In the details, Give the instance name(eg. Instance1).

Step 3: Click Instance Boot Source list and choose '**Boot from image**'

Step 4: Click Image name list and choose the image currently uploaded.

Step 5: Click launch.

Your VM will get created.

OPEN STACK INSTALLATION

\$ free -m - its shows the RAM Usage

\$ sudo adduser stack - password: stack

\$ sudo apt-get install sudo - for updating SU

\$ sudo -I - entering into the root user

echo "stack ALL=(ALL) NOPASSWD: ALL">> /etc/sudoers - it changes the stack from normal users to super user

#cat /etc/sudoers -To check last line as - (stack ALL =(ALL) NOPASSWD:ALL)

#exit - logout from root user

log off your machine and login as stack user

Prerequisite- to install git open all the ports

\$ sudo apt-get install git

\$ git clone https://git.openstack.org/openstack-dev/devstack

\$ ls - devstack should there

\$ cd devstack

\$ ls - it shows all the files in devstack

\$ls -l - longlisting the devstack files

\$ nano local.conf - paste in local.conf

[[local|localrc]]

FLOATING_RANGE=192.168.1.224/27

FIXED_RANGE=10.11.12.0/24

FIXED_NETWORK_SIZE=256

FLAT_INTERFACE=eth0

ADMIN_PASSWORD=root

DATABASE_PASSWORD=root

RABBIT_PASSWORD= root

SERVICE_PASSWORD= root

SERVICE_TOKEN= root save and quit

\$ ls -l - should be stack stack

\$./stack.sh - if error number 181 comes go to vi stack.sh add next to UMASK 022

FORCE=yes

\$ pwd

/home/stack/devstack

```

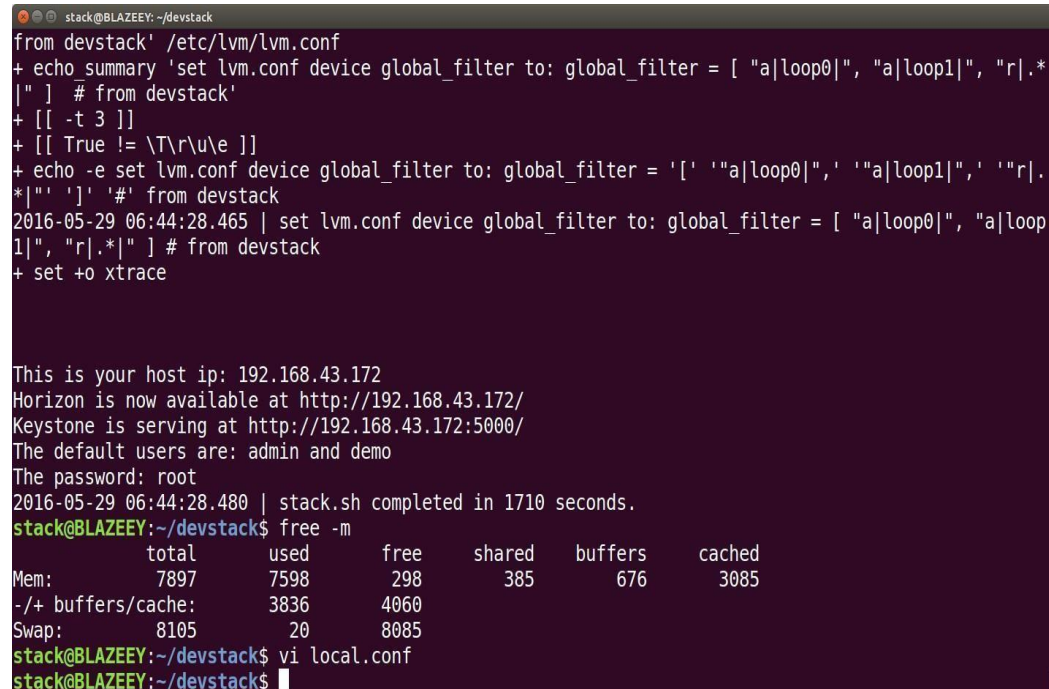
Devstack$ mv local.conf ../
$ cd ..
Stack$ rm -rf devstack/ /opt/stack/
$ git clone https://git.openstack.org/openstack-dev/devstack -b stable/kilo
$ cd devstack
$ ll
$ mv ../local.conf .
$ ls -l local.conf
$ ./stack.sh
$ nano stack.sh edit
#make sure unmask is sane
Add FORCE=yes save and exit
$ ./unstack.sh
$ ./clean.sh

```

Run DevStack:

```
$ ./stack.sh
```

Find User name, Password, IP address



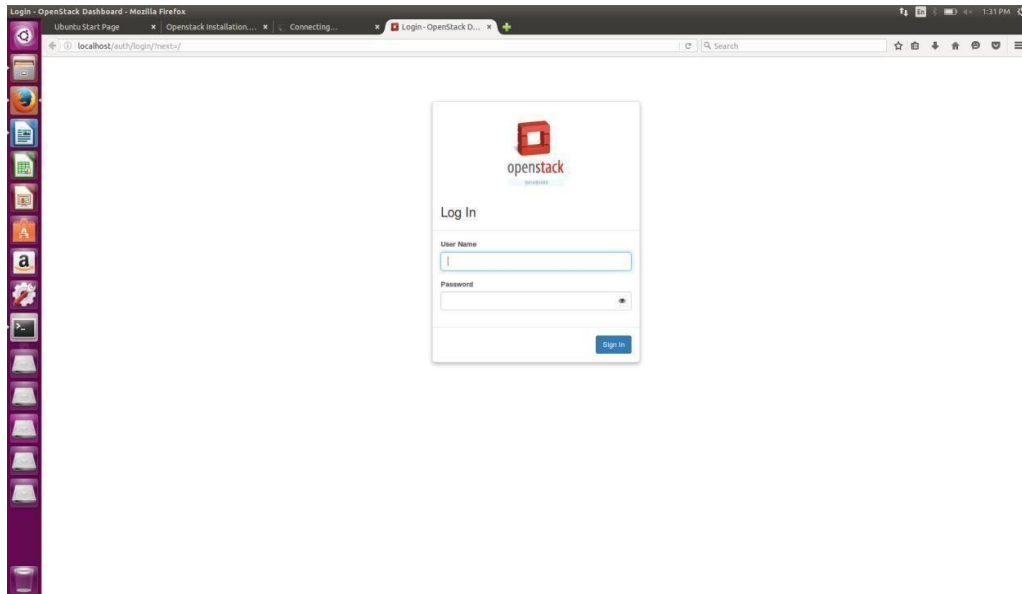
```

stack@BLAZEY: ~/devstack
from devstack' /etc/lvm/lvm.conf
+ echo summary 'set lvm.conf device global_filter to: global_filter = [ "a|loop0|", "a|loop1|", "r|.*
|" ] # from devstack'
+ [[ -t 3 ]]
+ [[ True != \T\r\u\e ]]
+ echo -e set lvm.conf device global_filter to: global_filter = '[' '"a|loop0|",' '"a|loop1|",' '"r|.
*|"' ']' '#' from devstack
2016-05-29 06:44:28.465 | set lvm.conf device global_filter to: global_filter = [ "a|loop0|", "a|loop
1|", "r|.*|" ] # from devstack
+ set +o xtrace

This is your host ip: 192.168.43.172
Horizon is now available at http://192.168.43.172/
Keystone is serving at http://192.168.43.172:5000/
The default users are: admin and demo
The password: root
2016-05-29 06:44:28.480 | stack.sh completed in 1710 seconds.
stack@BLAZEY:~/devstack$ free -m
              total        used        free      shared    buffers     cached
Mem:           7897         7598          298          385          676          3085
-/+ buffers/cache:        3836         4060
Swap:          8105           20         8085
stack@BLAZEY:~/devstack$ vi local.conf
stack@BLAZEY:~/devstack$

```

In browser give your machines ip <http://<ip Address>>



Re-Starting Openstack

\$./rejoin.sh

\$ ps -ef|grep devstack it shows all the processes running

End all the processes.

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the procedure to run the virtual machine of different configuration is executed successfully.

EX. NO: 03 DATE:23.7.22	FIND PROCEDURE TO ATTACH VIRTUAL BLOCK TO THE VIRTUAL MACHINE AND CHECK WHETHER IT HOLDS THE DATA EVEN AFTER THE RELEASE OF THE VIRTUAL MACHINE
--	--

AIM:

To write the procedure to attach virtual block to the virtual machine and check whether it holds the data even after the release of the virtual machine.

PROCEDURE:

Volumes are block storage devices that you attach to instances to enable persistent storage. You can attach a volume to a running instance or detach a volume and attach it to another instance at any time. You can also create a snapshot from or delete a volume. Only administrative users can create volume types.

Create a volume

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the Project tab, open the Compute tab and click Volumes category.
4. Click Create Volume.

In the dialog box that opens, enter or select the following values.

Volume Name: Specify a name for the volume.

Description: Optionally, provide a brief description for the volume.

Volume Source: Select one of the following options:

- o No source, empty volume: Creates an empty volume. An empty volume does not contain a file system or a partition table.
- o Image: If you choose this option, a new field for Use image as a source displays.

You can select the image from the list.

Volume: If you choose this option, a new field for Use volume as a source displays. You can select the volume from the list. Options to use a snapshot or a volume as the source for a volume are displayed only if there are existing snapshots or volumes.

Type: Leave this field blank.

Size (GB): The size of the volume in gibibytes (GiB).

Availability Zone: Select the Availability Zone from the list. By default, this value is set to the availability zone given by the cloud provider (for example, us-west or apac-south).

For some cases, it could be nova.

5. Click Create Volume.

The dashboard shows the volume on the Volumes tab.

On the *Project* tab, open the *Compute* tab and click the *Volumes* category.

4. Select the volume and click *Manage Attachments*.

5. Click *Detach Volume* and confirm your changes.

A message indicates whether the action was successful.

Create a snapshot from a volume

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.

3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.

4. Select a volume from which to create a snapshot.

5. In the *Actions* column, click *Create Snapshot*.

6. In the dialog box that opens, enter a snapshot name and a brief description.

7. Confirm your changes.

The dashboard shows the new volume snapshot in Volume Snapshots tab.

Edit a volume

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.

3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.

4. Select the volume that you want to edit.

5. In the *Actions* column, click *Edit Volume*.

6. In the *Edit Volume* dialog box, update the name and description of the volume.

7. Click *Edit Volume*.

Delete a volume

When you delete an instance, the data in its attached volumes is not deleted.

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.

3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.

Attach a volume to an instance

After you create one or more volumes, you can attach them to instances. You can attach a volume to one instance at a time.

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.

3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Select the volume to add to an instance and click *Manage Attachments*.
5. In the *Manage Volume Attachments* dialog box, select an instance.
6. Enter the name of the device from which the volume is accessible by the instance.
7. Click *Attach Volume*.

The dashboard shows the instance to which the volume is now attached and the device name.

You can view the status of a volume in the Volumes tab of the dashboard. The volume is either Available or In-Use.

Now you can log in to the instance and mount, format, and use the disk.

Detach a volume from an instance

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the *Project* tab, open the *Compute* tab and click *Volumes* category.
4. Select the check boxes for the volumes that you want to delete.
5. Click *Delete Volumes* and confirm your choice.

A message indicates whether the action was successful.

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the procedure to attach virtual block to the virtual machine and check whether it holds the data even after the release of the virtual machine is executed successful.

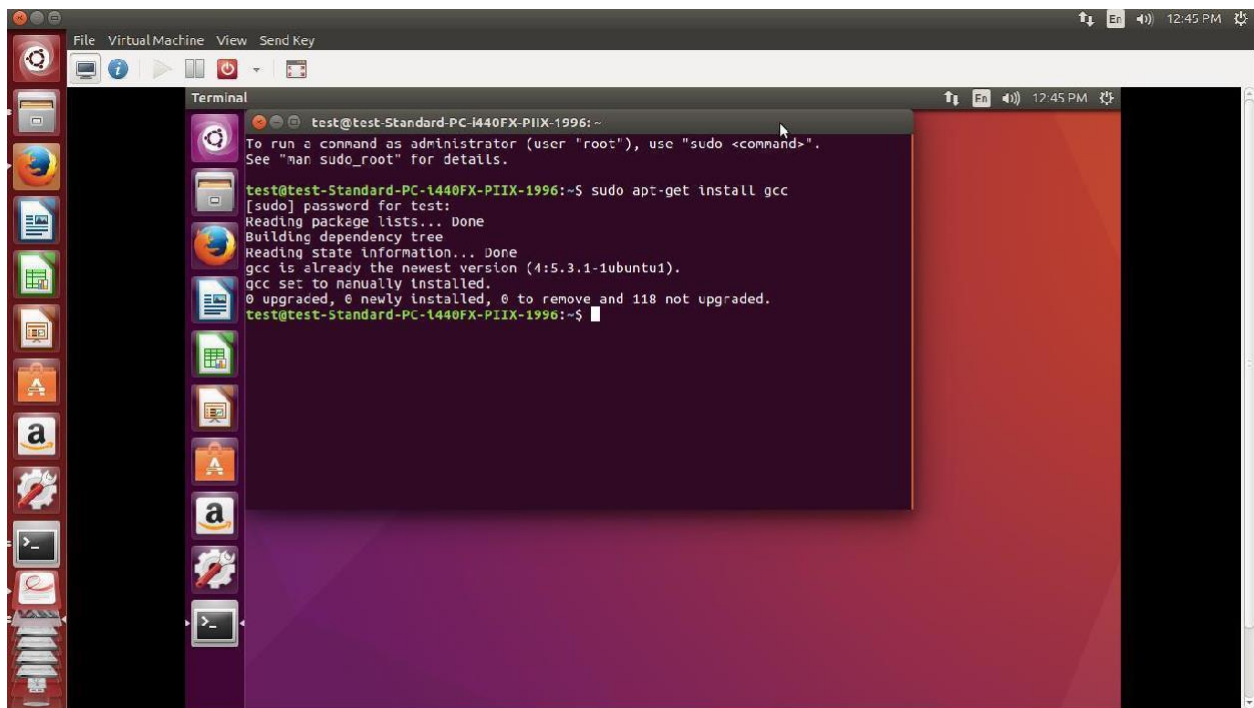
EX. NO: 04	INSTALL A C COMPILER IN THE VIRTUAL MACHINE AND
DATE:25.7.22	EXECUTE A SAMPLE PROGRAM

AIM:

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

PROCEDURE:

Step 1: To login into OS in VirtualBox



Step 2: To write and execute your own C Program in gcc compiler.

Install c compiler using commands.

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

PROGRAM

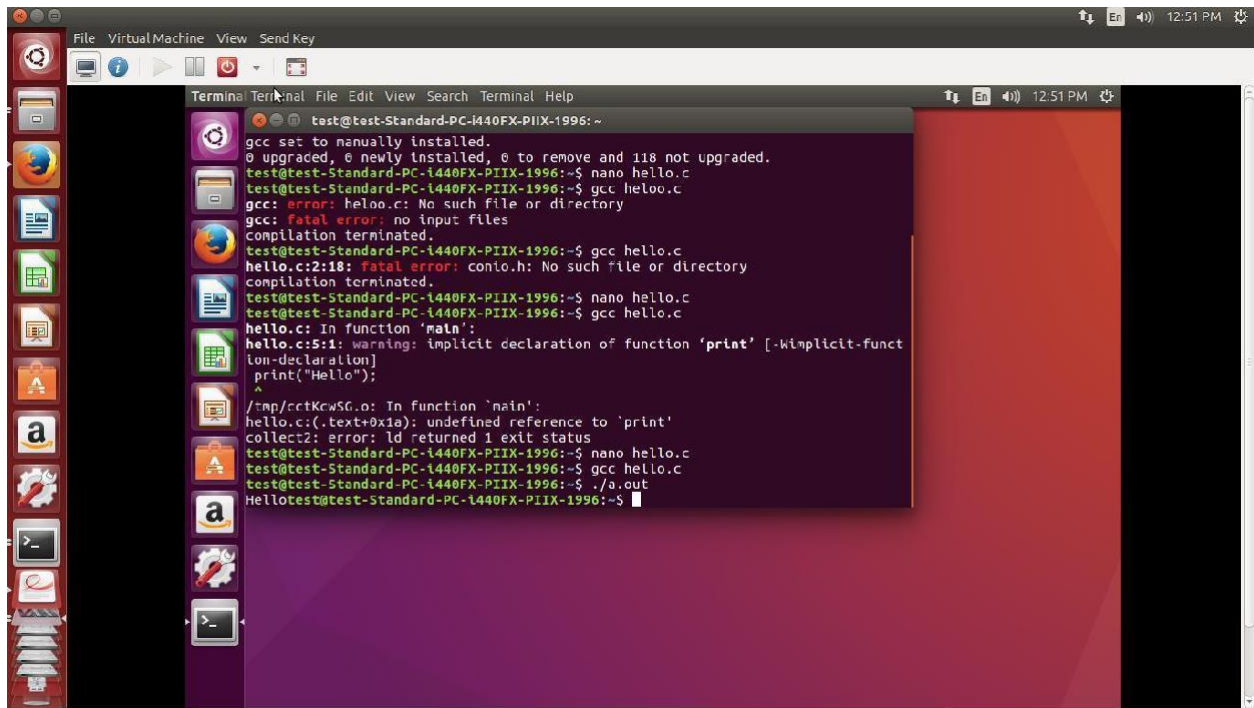
```
#include <stdio.h>
#include <string.h>
int main(){
    char string1[20];
    int i, length, flag=0;
    scanf("%s", string1);
    length = strlen(string1);
    for(i=0; i < length ; i++){
        if(string1[i] != string1[length-i-1]){
```



```

    flag = 1;
    break;
}
}
if (flag) {
    printf("%s is not a palindrome", string1);
}
else {
    printf("%s is a palindrome", string1);
}
return 0;
}

```



DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the installation of a C compiler and execution of a C program in the virtual machine is executed successfully.

EX. NO: 05	SHOW THE VIRTUAL MACHINE MIGRATION BASED ON THE CERTAIN CONDITION FROM ONE NODE TO THE OTHER
DATE:1.8.22	

AIM:

To learn virtual machine migration based on the certain condition from one node to the other.

PROCEDURE:

To demonstrate virtual machine migration, two machines must be configured in one cloud.

MIGRATION LIMITATIONS

Openstack has two commands specific to virtual machine migration:

- ☐ nova migrate \$UUID
- ☐ nova live-migration \$UUID \$COMPUTE-HOST

The nova migrate command shuts down an instance to move it to another hypervisor.

- ☐ The instance is down for a period of time and sees this as a regular shutdown.
- ☐ It is not possible to specify the compute host you want to migrate the instance to.

(Read on to see how you can do that the dirty way).

- ☐ This command does not require shared storage, the migrations can take a long time.
- ☐ The Openstack cluster chooses the target hypervisor machine based on the free resources and availability.
- ☐ The migrate command works with any type of instance.
- ☐ The VM clock has no issues.

The nova live-migration command has almost no instance downtime.

- ☐ The instance is suspended and does not see this as a shutdown.
- ☐ The live-migration lets you specify the compute host you want to migrate to, however with some limitations.

This requires shared storage, instances without a configdrive when block storage is used, or volume-backed instances.

- ☐ The migration fails if there are not enough resources on the target hypervisor
- ☐ The VM clock might be off.

Here are some examples when to use which option:

- ☐ If it is important to choose the compute host or to have very little downtime you need to use `nova live-migration` command.
- ☐ If you don't want to choose the compute host, or you have a configdrive enabled, you need to use the `nova migrate` command.
- ☐ If you need to specify the compute host and you have a configdrive enabled, you need to manually migrate the machine, or use a dirty trick to fool nova migrate.

Hypervisor Capacity

Before you do a migration, check if the hypervisor host has enough free capacity for the VM you want to migrate:

```
nova host-describe compute-30
```

Example output:

```
+-----+-----+-----+-----+
| HOST | PROJECT | cpu | memory_mb | disk_gb |
+-----+-----+-----+-----+
| compute-30 | (total) | 64 | 512880 | 5928 |
| compute-30 | (used_now) | 44 | 211104 | 892 |
| compute-30 | (used_max) | 44 | 315568 | 1392 |
| compute-30 | 4[...]0288 | 1 | 512 | 20 |
| compute-30 | 4[...]0194 | 20 | 4506 | 62 |
```

In this table, the first row shows the total amount of resources available on the physical server. The second line shows the currently used resources. The third line shows the maximum used resources. The fourth line and below shows the resources available for each project.

If the VM flavor fits on this hypervisor, continue on with the manual migration. If not, free up some resources or choose another compute server. If the hypervisor node lacks enough capacity, the migration will fail.

(Live) migration with nova live-migration

The live-migration command works with the following types of vm's/storage:

- ☐ Shared storage: Both hypervisors have access to shared storage.

❑ Block storage: No shared storage is required. Instances are backed by image based root disks. Incompatible with read-only devices such as CD-ROMs and Configuration Drive (config_drive).

❑ Volume storage: No shared storage is required. Instances are backed by iSCSI volumes rather than ephemeral disk.

The live-migration command requires the same CPU on both hypervisors. It is possible to set a generic CPU for the VM's, or a generic set of CPU features. This however does not work on versions lower than Kilo due to a bug where Nova compares the actual CPU instead of the virtual CPU. In my case, all the hypervisor machines are the same, lucky me. This is fixed in Kilo or later.

On versions older than Kilo, the Compute service does not use libvirt's live migration functionality by default, therefore guests are suspended before migration and might experience several minutes of downtime. This is because there is a risk that the migration process will never end. This can happen if the guest operating system uses blocks on the disk faster than they can be migrated. To enable true live migration using libvirt's migrate functionality, see the Openstack documentation linked below.

Shared storage / Volume backed instances

A live-migration is very simple. Use the following command with an instance UUID and the name of the compute host:

nova live-migration \$UUID \$COMPUTE-HOST

If you have shared storage, or if the instance is volume backed, this will send the instances memory (RAM) content over to the destination host. The source hypervisor keeps track of which memory pages are modified on the source while the transfer is in progress. Once the initial bulk transfer is complete, pages changed in the meantime are transferred again. This is done repeatedly with (ideally) ever smaller increments.

As long as the differences can be transferred faster than the source VM dirties memory pages, at some point the source VM gets suspended. Final differences are sent to the target host and an identical machine started there. At the same time the virtual network infrastructure takes care of all traffic being directed to the new virtual machine. Once the replacement machine is running, the suspended source instance is deleted. Usually the actual handover takes place so quickly and seamlessly that all but very time sensitive applications ever notice anything.

You can check this by starting a ping to the VM you are live-migrating. It will stay

online and when the VM is suspended and resumed on the target hypervisor, the ping responses will take a bit longer.

Block based storage (--block-migrate)

The process is almost exactly the same as described above. There is one extra step however. Before the memory contents is sent the disk content is copied over, without downtime. When the VM is suspended, both the memory contents and the disk contents (difference to the earlier copy) are sent over. The suspend action takes longer and might be noticeable as downtime. The --block-migrate option is incompatible with read only devices such as ISO CD/DVD drives and the Config Drive.

Migration with nova migrate

The nova migrate command shuts down an instance, copies over the disk to a hypervisor with enough free resources, starts it up there and removes it from the source hypervisor.

The VM is shut down and will be down as long as the copying. With a migrate, the Openstack cluster chooses an compute-service enabled hypervisor with the most resources available. This works with any type of instance, with any type of backend storage.

A migrate is even simpler than a live-migration. Here's the syntax:

```
nova migrate $UUID
```

This is perfect for instances that are part of a clustered service, or when you have scheduled and communicated downtime for that specific VM. The downtime is dependent on the size of the disk and the speed of the (storage) network. rsync over ssh is used to copy the actual disk, you can test the speed yourself with a few regular rsync tests, and combine that with the disksize to get an indication of the migration downtime.

Migrating to a specific compute node, the dirty way

As seen above, we cannot migrate virtual machines to a specific compute node if the compute node does not have shared storage and the virtual machine has a configdrive enabled. You can force the Openstack cluster to choose a specific hypervisor by disabling the nova-compute service on all the other hypervisors. The VM's will keep running on there, only new virtual machines and migrations are not possible on those hypervisors.

If you have a lot of creating and removing of machines in your Openstack Cloud, this might be a bad idea. If you use (Anti) Affinity Groups, vm's created in there will also fail to start depending on the type of Affinity Group. Therefore, use this option with caution. If we have 5 compute nodes, compute-30 to compute-34 and we want to migrate the machine to compute-34, we need to disable the nova-compute service on all other hypervisors.

First check the state of the cluster:

nova service-list --binary nova-compute # or nova-conductor, nova-cert, novaconsoleauth, nova-scheduler

Example output:

```
+__+____+____+____+____+.....+.....
      +.....+
| Id | Binary | Host | Zone | Status | State | Updated_at
| Disabled Reason |
+__+____+____+____+____+.....+.....
      +.....+
| 7 | nova-compute | compute-30 | OS1 | enabled | up | 2015-06-
13T17:04:27.000000 | - |
| 8 | nova-compute | compute-31 | OS2 | enables | up | 2015-06-
13T17:02:49.000000 | - |
| 9 | nova-compute | compute-32 | OS2 | enabled | up | 2015-06-
13T17:02:50.000000 | None |
| 10 | nova-compute | compute-33 | OS2 | enabled | up | 2015-06-
13T17:02:50.000000 | - |
| 11 | nova-compute | compute-34 | OS1 | disabled | up | 2015-06-
13T17:02:49.000000 | Migrations Only |
      +__+.....+.....+.....+.....
```

In this example we have 5 compute nodes, of which one is disabled with reason Migrations Only. In our case, before we started migrating we have enabled nova compute on that hypervisor and disabled it on all the other hypervisors:

```
nova service-disable compute-30 nova-compute --reason 'migration to specific
hypervisor the dirty way'
```

```
nova service-disable compute-31 nova-compute --reason 'migration to specific
hypervisor the dirty way'
```

etc...

Now execute the nova migrate command. Since you've disabled all compute hypervisors except the target hypervisor, that one will be used as migration target. All new virtual machines created during the migration will also be spawned on that specific hypervisor. When the migration is finished, enable all the other compute nodes:

```
nova service-enable compute-30 nova-compute
```

```
nova service-enable compute-31 nova-compute
```

etc...

In our case, we would disable the compute-34 because it is for migrations only. This is a bit dirty and might cause problems if you have monitoring on the cluster state or spawn a lot of machines all the time.

Manual migration to a specific compute node

As seen above, we cannot migrate virtual machines to a specific compute node if the compute node does not have shared storage and the virtual machine has a configdrive enabled. Since Openstack is just a bunch of wrappers around native Linux tools, we can manually migrate the machine and update the Nova database afterwards.

Do note that this part is specific to the storage you use. In this example we use local storage (or, a local folder on an NFS mount not shared with other compute nodes) and image-backed instances. In my case, I needed to migrate an image-backed block storage instance to a non-shared storage node, but the instance had a configdrive enabled. Disabling the compute service everywhere is not an option, since the cluster was getting about a hundred new VM's every 5 minutes and that would overload the hypervisor node.

This example manually migrates a VM from compute-30 to compute-34. These nodes are in the same network and can access one another via SSH keys based on their hostname.

Shut down the VM first:

```
nova stop $VM_UUID
```

Also detach any volumes:

```
nova volume-detach $VM_UUID $VOLUME_UUID
```

Use the nova show command to see the specific hypervisor the VM is running on:

```
nova show UUID | grep hypervisor
```

Example output:

```
| OS-EXT-SRV-ATTR:hypervisor_hostname | compute-30 |
```

Login to that hypervisor via SSH. Navigate to the folder where this instance is located, in our case, /var/lib/nova-compute/instances/\$UUID.

The instance is booted from an image based root disk, named disk. qemu in our case diffs the root disk from the image the VM was created from. Therefore the new hypervisor also needs that backing image. Find out which file is the backing image:

```
cd /var/lib/nova-compute/instances/UUID/
```

```
qemu-img info disk # disk is the filename of the instance root disk
```

Example output:

```
image: disk
```

file format: qcow2

virtual size: 32G (34359738368 bytes)

disk size: 1.3G

cluster_size: 65536

backing file: /var/lib/nova-compute/instances/_base/d00[...]61

Format specific information:

compat: 1.1

lazy refcounts: false

The file /var/lib/novacompute/

instances/_base/d004f7f8d3f79a053fad5f9e54a4aed9e2864561 is the backing

disk. Note that the long filename is not a UUID but a checksum of the specific image version.

In my case it is a raw disk:

qemu-img info /var/lib/nova-compute/instances/_base/d00[...]61

Example output:

image: /var/lib/nova-compute/instances/_base/d00[...]61

file format: raw

virtual size: 8.0G (8589934592 bytes)

disk size: 344M

Check the target hypervisor for the existence of that image. If it is not there, copy that file to the target hypervisor first:

```
rsync -r --progress /var/lib/nova-compute/instances/_base/d00[...]61 -e ssh
compute-34:/var/lib/nova-compute/instances/_base/d00[...]61
```

On the target hypervisor, set the correct permissions:

```
chown nova:nova /var/lib/nova-compute/instances/_base/d00[...]61
```

Copy the instance folder to the new hypervisor:

```
cd /var/lib/nova-compute/instances/
rsync -r --progress $VM_UUID -e ssh compute-34:/var/lib/nova-compute/instances/
```

Set the correct permissions on the folder on the target hypervisor:

```
chown nova:nova /var/lib/nova-compute/instances/$VM_UUID
```

```
chown nova:nova /var/lib/nova-compute/instances/$VM_UUID/disk.info
```

```
chown nova:nova /var/lib/nova-compute/instances/libvirt.xml
```

```
chown libvirt:kvm /var/lib/nova-compute/instances/$VM_UUID/console.log
```

```
chown libvirt:kvm /var/lib/nova-compute/instances/$VM_UUID/disk
```

```
chown libvirt:kvm /var/lib/nova-compute/instances/$VM_UUID/disk.config
```


If you use other usernames and groups, change those in the command.

Log in to your database server. In my case that is a MySQL Galera cluster. Start up a MySQL command prompt in the novadatabase

```
mysql nova
```

Execute the following command to update the nova database with the new hypervisor for this VM:

```
update instances set node='compute-34', host=node where uuid='$VM_UUID';
```

This was tested on an IceHouse database scheme, other versions might require other queries.

Use the nova show command to see if the new hypervisor is set. If so, start the VM:

```
nova start $VM_UUID
```

Attach any volumes that were detached earlier:

```
nova volume-attach $VM_UUID $VOLUME_UUID
```

Use the console to check if it all works:

```
nova get-vnc-console $VM_UUID novnc
```

Do note that you must check the free capacity yourself. The VM will work if there is not enough capacity, but you do run in to weird issues with the hypervisor like bad performance or killed processes (OOM's).

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the virtual machine migration based on the certain condition from one node to the other was executed successfully.

EX. NO: 06	FIND PROCEDURE TO INSTALL STORAGE CONTROLLER
DATE:8.8.22	AND INTERACT WITH IT

AIM:

To find procedure to install storage controller and interact with it.

PROCEDURE:**OpenStack Block Storage**

The OpenStack Block Storage service (cinder) adds persistent storage to a virtual machine. Block Storage provides an infrastructure for managing volumes, and interacts with OpenStack Compute to provide volumes for instances. The service also enables management of volume snapshots, and volume types. The Block Storage service consists of the following components:

cinder-api

Accepts API requests, and routes them to the cinder-volume for action.

cinder-volume

Interacts directly with the Block Storage service, and processes such as the cinder-scheduler. It also interacts with these processes through a message queue. The cinder-volume service responds to read and write requests sent to the Block Storage service to maintain state. It can interact with a variety of storage providers through a driver architecture.

cinder-scheduler daemon

Selects the optimal storage provider node on which to create the volume. A similar component to the nova-scheduler.

Messaging queue

Routes information between the Block Storage processes.

Install and configure controller node

This section describes how to install and configure the Block Storage service, code-named cinder, on the controller node. This service requires at least one additional storage node that provides volumes to instances.

To configure prerequisites

Before you install and configure the Block Storage service, you must create a database, service credentials, and API endpoints.

1. To create the database, complete these steps:

a. Use the database access client to connect to the database server as

the root user:

```
$ mysql -u root -p
```

2. Create the cinder database:

```
CREATE DATABASE cinder;
```

a. Grant proper access to the cinder database:

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
IDENTIFIED BY 'CINDER_DBPASS';
```

```
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
IDENTIFIED BY 'CINDER_DBPASS';
```

Replace CINDER_DBPASS with a suitable password.

b. Exit the database access client.

3. Source the admin credentials to gain access to admin-only CLI commands:

```
$ source admin-openrc.sh
```

4. To create the service credentials, complete these steps:

a. Create a cinder user:

```
$ keystone user-create --name cinder --pass CINDER_PASS
```

```
+.....+.....+
| Property | Value |
+.....+.....+
| email | |
| enabled | True |
| id | 881ab2de4f7941e79504a759a83308be |
| name | cinder |
| username | cinder |
```

```
+.....+.....+
```

Replace CINDER_PASS with a suitable password.

b. Add the admin role to the cinder user:

```
$ keystone user-role-add --user cinder --tenant service --role admin
```

c. Create the cinder service entities:

```
$ keystone service-create --name cinder --type volume \
--description "OpenStack Block Storage"
```

```
+.....+.....+
```

```

| Property | Value |
+-----+-----+
| description | OpenStack Block Storage |
| enabled | True |
| id | 1e494c3e22a24baafcaf777d4d467eb |
| name | cinder |
| type | volume |
+-----+-----+
$ keystone service-create --name cinderv2 --type volumev2 \
--description "OpenStack Block Storage"

```

```

+-----+-----+
| Property | Value |
+-----+-----+
| description | OpenStack Block Storage |
| enabled | True |
| id | 16e038e449c94b40868277f1d801edb5 |
| name | cinderv2 |
| type | volumev2 |

```

```

+-----+-----+
4. Create the Block Storage service API endpoints:

```

```

$ keystone endpoint-create \
--service-id $(keystone service-list | awk '/ volume / {print $2}') \
--publicurl http://controller:8776/v1/$(tenant_id)s \
--internalurl http://controller:8776/v1/$(tenant_id)s \
--adminurl http://controller:8776/v1/$(tenant_id)s \
--region regionOne

```

```

+-----+-----+
| Property | Value |
+-----+-----+
| adminurl | http://controller:8776/v1/$(tenant_id)s |
| id | d1b7291a2d794e26963b322c7f2a55a4 |
| internalurl | http://controller:8776/v1/$(tenant_id)s |
| publicurl | http://controller:8776/v1/$(tenant_id)s |
| region | regionOne |

```

```
| service_id | 1e494c3e22a24baafcaf777d4d467eb |
+-----+-----+
$ keystone endpoint-create \
--service-id $(keystone service-list | awk '/ volumev2 / {print $2}') \
--publicurl http://controller:8776/v2/%(tenant_id)s \
--internalurl http://controller:8776/v2/%(tenant_id)s \
--adminurl http://controller:8776/v2/%(tenant_id)s \
--region regionOne
+-----+-----+
| Property | Value |
+-----+-----+
| adminurl | http://controller:8776/v2/%(tenant_id)s |
| id | 097b4a6fc8ba44b4b10d4822d2d9e076 |
| internalurl | http://controller:8776/v2/%(tenant_id)s |
| publicurl | http://controller:8776/v2/%(tenant_id)s |
| region | regionOne |
| service_id | 16e038e449c94b40868277f1d801edb5 |
+-----+-----+
```

To install and configure Block Storage controller components

1. Install the packages:

```
# apt-get install cinder-api cinder-scheduler python-cinderclient
```

2. Edit the /etc/cinder/cinder.conf file and complete the following actions:

a. In the [database] section, configure database access:

```
[database]
```

```
...
```

```
connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

Replace *CINDER_DBPASS* with the password you chose for the Block Storage database.

b. In the [DEFAULT] section, configure RabbitMQ message broker access:

```
[DEFAULT]
```

```
...
```

```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
...
```

```
auth_uri = http://controller:5000/v2.0
```

```
identity_uri = http://controller:35357
```

```
admin_tenant_name = service
```

```
admin_user = cinder
```

```
admin_password = CINDER_PASS
```

3. Replace CINDER_PASS with the password you chose for the cinder user in the Identity service.

4. d. In the [DEFAULT] section, configure the my_ip option to use the management interface IP address of the controller node:

```
[DEFAULT]
```

```
...
```

```
my_ip = 10.0.0.11
```

e. (Optional) To assist with troubleshooting, enable verbose logging in the [DEFAULT] section:

```
[DEFAULT]
```

```
...
```

```
verbose = True
```

3. Populate the Block Storage database:

```
# su -s /bin/sh -c "cinder-manage db sync" cinder
```

To finalize installation

1. Restart the Block Storage services:

```
# service cinder-scheduler restart
```

```
# service cinder-api restart
```

2. By default, the Ubuntu packages create an SQLite database.

Because this configuration uses a SQL database server, you can remove the SQLite database file:

```
# rm -f /var/lib/cinder/cinder.sqlite
```

Install and configure a storage node

This section describes how to install and configure storage nodes for the Block Storage service. For simplicity, this configuration references one storage node with an empty local block

storage device /dev/sdb that contains a suitable partition table with one partition /dev/sdb1 occupying the entire device. The service provisions logical volumes on this device using the LVM driver and provides them to instances via iSCSI transport. You can follow

these instructions with minor modifications to horizontally scale your environment with additional storage nodes.

To configure prerequisites

You must configure the storage node before you install and configure the volume service on it. Similar to the controller node, the storage node contains one network interface on the management network. The storage node also needs an empty block storage device of suitable size for your environment.

1. Configure the management interface:

IP address: 10.0.0.41

Network mask: 255.255.255.0 (or /24)

Default gateway: 10.0.0.1

2. Set the hostname of the node to *block1*.

3. Copy the contents of the `/etc/hosts` file from the controller node to the storage node and add the following to it:

```
# block1
```

```
10.0.0.41 block1
```

Also add this content to the `/etc/hosts` file on all other nodes in your environment.

4. Install and configure NTP using the instructions in the section called “Other nodes”.

5. Install the LVM packages:

```
# apt-get install lvm2
```

Create the LVM physical volume `/dev/sdb1`:

```
# pvcreate /dev/sdb1
```

Physical volume `"/dev/sdb1"` successfully created

7. Create the LVM volume group `cinder-volumes`:

```
# vgcreate cinder-volumes /dev/sdb1
```

Volume group `"cinder-volumes"` successfully created

The Block Storage service creates logical volumes in this volume group.

Only instances can access Block Storage volumes. However, the underlying operating system manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the `/dev` directory for block storage devices that contain volumes. If tenants use LVM on

their volumes, the scanning tool detects these volumes and attempts to cache them which can

cause a variety of problems with both the underlying operating system and tenant volumes.

You

must reconfigure LVM to scan only the devices that contain the cinder-volume volume group.

Edit the `/etc/lvm/lvm.conf` file and complete the following actions:

a. In the devices section, add a filter that accepts the `/dev/sdb` device and rejects all other devices:

```
devices {
```

```
...
```

```
filter = [ "a/sdb/", "r/.*/"]
```

Each item in the filter array begins with a for *accept* or r for *reject* and includes a regular expression for the device name. The array must end with `r/.*/` to reject any remaining devices.

You can use the `vgs -vvvv` command to test filters.

Install and configure Block Storage volume components

1. Install the packages:

```
# apt-get install cinder-volume python-mysqldb
```

2. Edit the `/etc/cinder/cinder.conf` file and complete the following actions:

3. In the `[database]` section, configure database access:

```
[database]
```

```
...
```

```
connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

Replace `CINDER_DBPASS` with the password you chose for the Block Storage database.

a. In the `[DEFAULT]` section, configure RabbitMQ message broker access:

```
[DEFAULT]
```

```
...
```

```
rpc_backend = rabbit
```

```
rabbit_host = controller
```

```
rabbit_password = RABBIT_PASS
```

Replace `RABBIT_PASS` with the password you chose for the guest account in RabbitMQ.

c. In the `[DEFAULT]` and `[keystone_authtoken]` sections, configure Identity service access:

```
[DEFAULT]
```

```
...
```



```
auth_strategy = keystone
```

```
[keystone_authtoken]
```

```
...
```

```
auth_uri = http://controller:5000/v2.0
```

```
identity_uri = http://controller:35357
```

```
admin_tenant_name = service
```

```
admin_user = cinder
```

```
admin_password = CINDER_PASS
```

Replace *CINDER_PASS* with the password you chose for the cinder user in the Identity service.

d. In the [DEFAULT] section, configure the my_ip option:

```
[DEFAULT]
```

```
...
```

```
my_ip = MANAGEMENT_INTERFACE_IP_ADDRESS
```

Replace *MANAGEMENT_INTERFACE_IP_ADDRESS* with the IP address of the management

network interface on your storage node, typically 10.0.0.41 for the first node in the example architecture.

e. In the [DEFAULT] section, configure the location of the Image Service:

```
[DEFAULT]
```

```
...
```

```
glance_host = controller
```

f. (Optional) To assist with troubleshooting, enable verbose logging in the [DEFAULT] section:

```
[DEFAULT]
```

```
...
```

```
verbose = True
```

To finalize installation

1. Restart the Block Storage volume service including its dependencies:

```
# service tgt restart
```

```
# service cinder-volume restart
```

2. By default, the Ubuntu packages create an SQLite database. Because this configuration uses a SQL

database server, remove the SQLite database file:

```
# rm -f /var/lib/cinder/cinder.sqlite
```

Verify operation

This section describes how to verify operation of the Block Storage service by creating a volume

Source the admin credentials to gain access to admin-only CLI commands:

```
$ source admin-openrc.sh
```

2. List service components to verify successful launch of each process:

\$ cinder service-list

Binary	Host	Zone	Status	State
--------	------	------	--------	-------

Updated_at | Disabled Reason |

[illegible]

```
| cinder-scheduler | controller | nova | enabled | up | 2014-10-18T01:30:54.000000 | None |
```

```
| cinder-volume | block1 | nova | enabled | up | 2014-10-18T01:30:57.000000 | None |
```

The diagram illustrates the recursive step in the proof. It shows a large rectangle divided into two parts by a vertical dashed line. The left part is a 2x1 grid, and the right part is a 2x(N-1) grid. The 2x1 grid is further divided into two 1x1 squares. The 2x(N-1) grid is divided into two 1x(N-1) rectangles. The top row of the 2x(N-1) grid is labeled with a '+' sign at the start, followed by a '+' sign at the end of the first 1x(N-1) rectangle, and then a '+' sign at the end of the second 1x(N-1) rectangle. The bottom row of the 2x(N-1) grid is labeled with a '+' sign at the start, followed by a '+' sign at the end of the first 1x(N-1) rectangle, and then a '+' sign at the end of the second 1x(N-1) rectangle.

3. Source the demo tenant credentials to perform the following steps as a nonadministrative tenant:

```
$ source demo-openrc.sh
```

4. Create a 1 GB volume:

```
$ cinder create --display-name demo-volume1 1
```

Property	Value
----------	-------

| attachments | [] |

| availability_zone | nova |

```
| bootable | false |
```

```
| created_at | 2014-10-14T23:11:50.870239 |
```

| display_description | None |

```
| display_name | demo-volume1 |
```

```
| encrypted | False |
| id | 158bea89-07db-4ac2-8115-66c0d6a4bb48 |
| metadata | { } |
| size | 1 |
| snapshot_id | None |
| source_volid | None |
| status | creating |
| volume_type | None |
+ ..... + .....
```

5. Verify creation and availability of the volume:

\$ cinder list

```
.....+ .....+ .....+ .....+
.....+ .....+ .....+
| ID | Status | Display Name | Size |
Volume Type | Bootable | Attached to |
+ .....+ .....+ .....+ .....+-
```

```
| 158bea89-07db-4ac2-8115-66c0d6a4bb48 | available | demo-volume1 | 1 |
None | false | |
```

```
+ .....+ .....+ .....+ .....+-
```

Your OpenStack environment now includes Block Storage.

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the procedure to install storage controller and interact with openstack service is executed successfully.

EX. NO: 07	FIND PROCEDURE TO SET UP THE ONE NODE HADOOP CLUSTER
DATE:22.8.22	

AIM:

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

PROCEDURE:

1) Installing Java

Hadoop is a framework written in Java for running applications on large clusters of commodity hardware. Hadoop needs Java 6 or above to work.

Step 1: Download tar and extract

Download Jdk tar.gz file for linux-62 bit, extract it into “/usr/local”

```
# cd /opt
```

```
# sudo tar xvpzf /home/itadmin/Downloads/jdk-8u5-linux-x64.tar.gz
```

```
# cd /opt/jdk1.8.0_05
```

Step 2: Set Environments

☐ Open the “/etc/profile” file and Add the following line as per the version

☐ Set a environment for Java

☐ Use the root user to save the /etc/profile or use gedit instead of vi .

☐ The 'profile' file contains commands that ought to be run for login shells

```
# sudo vi /etc/profile
```

```
--insert JAVA_HOME
```

```
JAVA_HOME=/opt/jdk1.8.0_05
```

```
--in PATH variable just append at the end of the line
```

```
PATH=$PATH:$JAVA_HOME/bin
```

```
--Append JAVA_HOME at end of the export statement
```

```
export PATH JAVA_HOME
```

save the file using by pressing “Esc” key followed by :wq!

Step 3: Source the /etc/profile

```
# source /etc/profile
```

Step 4: Update the java alternatives

1. By default OS will have a openjdk. Check by “java -version”. You will be prompt “openJDK”

2. If you also have openjdk installed then you'll need to update the java alternatives:

3. If your system has more than one version of Java, configure which one your system causes by entering the following command in a terminal window

4. By default OS will have a open jdk. Check by “java -version”. You will be prompt “JavaHotSpot(TM) 64-Bit Server”

```
# update-alternatives --install "/usr/bin/java" java "/opt/jdk1.8.0_05/bin/java" 1
```

```
# update-alternatives --config java
```

```
--type selection number:
```

```
# java -version
```

2) configure ssh

☐ Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine if you want to use Hadoop on it (which is what we want to do in this exercise).

For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost

The need to create a Password-less SSH Key generation based authentication is so that the master node can then login to slave nodes (and the secondary node) to start/stop them easily without any delays for authentication

☐ If you skip this step, then have to provide password

Generate an SSH key for the user. Then Enable password-less SSH access to yo

```
sudo apt-get install openssh-server
```

--You will be asked to enter password,

```
root@abc []# ssh localhost
```

```
root@abc[]# ssh-keygen
```

```
root@abc[]# ssh-copy-id -i localhost
```

--After above 2 steps, You will be connected without password,

```
root@abc[]# ssh localhost
```

```
root@abc[]# exit
```

3) Hadoop installation

☐ Now Download Hadoop from the official Apache, preferably a stable release version of Hadoop 2.7.x and extract the contents of the Hadoop package to a location of your choice.

☐ For example, choose location as “/opt/”

Step 1: Download the tar.gz file of latest version Hadoop (hadoop-2.7.x) from the official site

.

Step 2: Extract (untar) the downloaded file from this commands to /opt/bigdata

```
root@abc[]# cd /opt
```

```
root@abc[/opt]# sudo tar xvpzf /home/itadmin/Downloads/hadoop-2.7.0.tar.gz
```

```
root@abc[/opt]# cd hadoop-2.7.0/
```

Like java, update Hadop environment variable in /etc/profile

```
# sudo vi /etc/profile
```

```
#--insert HADOOP_PREFIX
```

```
HADOOP_PREFIX=/opt/hadoop-2.7.0
```

```
#--in PATH variable just append at the end of the line
```

```
PATH=$PATH:$HADOOP_PREFIX/bin
```

```
#--Append HADOOP_PREFIX at end of the export statement
```

```
export PATH JAVA_HOME HADOOP_PREFIX
```

save the file using by pressing “Esc” key followed by :wq!

Step 3: Source the /etc/profile

```
# source /etc/profile
```

Verify Hadoop installation

```
# cd $HADOOP_PREFIX
```

```
# bin/hadoop version
```

3.1) Modify the Hadoop Configuration Files

□ In this section, we will configure the directory where Hadoop will store its configuration files, the network ports it listens to, etc. Our setup will use Hadoop Distributed File System,(HDFS), even though we are using only a single local machine.

□ Add the following properties in the various hadoop configuration files which is available under \$HADOOP_PREFIX/etc/hadoop/

□ core-site.xml, hdfs-site.xml, mapred-site.xml & yarn-site.xml

Update Java, hadoop path to the Hadoop environment file

```
# cd $HADOOP_PREFIX/etc/hadoop
```

```
# vi hadoop-env.sh
```

Paste following line at beginning of the file

```
export JAVA_HOME=/usr/local/jdk1.8.0_05
```

```
export HADOOP_PREFIX=/opt/hadoop-2.7.0
```

Modify the core-site.xml

```
# cd $HADOOP_PREFIX/etc/hadoop
```

```
# vi core-site.xml
```

Paste following between <configuration> tags

```
<configuration>
```

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

Modify the hdfs-site.xml

```
# vi hdfs-site.xml
```

Paste following between <configuration> tags

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

YARN configuration - Single Node

modify the mapred-site.xml

```
# cp mapred-site.xml.template mapred-site.xml
# vi mapred-site.xml
```

Paste following between <configuration> tags

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

Modiy yarn-site.xml

```
# vi yarn-site.xml
```

Paste following between <configuration> tags

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
</configuration>
```

Formatting the HDFS file-system via the NameNode

□ The first step to starting up your Hadoop installation is formatting the Hadoop files system which is implemented on top of the local file system of our “cluster” which includes only our local machine. We need to do this the first time you set up a Hadoop cluster.

□ Do not format a running Hadoop file system as you will lose all the data currently in the cluster (in HDFS)

```
root@abc[]# cd $HADOOP_PREFIX
```

```
root@abc[]# bin/hadoop namenode -format
```

Start NameNode daemon and DataNode daemon: (port 50070)

```
root@abc[]# sbin/start-dfs.sh
```

To know the running daemons jut type jps or /usr/local/jdk1.8.0_05/bin/jps

Start ResourceManager daemon and NodeManager daemon: (port 8088)

```
root@abc[]# sbin/start-yarn.sh
```

To stop the running process

```
root@abc[]# sbin/stop-dfs.sh
```

To know the running daemons jut type jps or /usr/local/jdk1.8.0_05/bin/jps

Start ResourceManager daemon and NodeManager daemon: (port 8088)

```
root@abc[]# sbin/stop-yarn.sh
```

Make the HDFS directories required to execute MapReduce jobs:

```
$ bin/hdfs dfs -mkdir /user
```

```
$ bin/hdfs dfs -mkdir /user/mit
```

□ Copy the input files into the distributed filesystem:

```
$ bin/hdfs dfs -put <input-path>/* /input
```

□ Run some of the examples provided:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.5.1.jar
```

```
grep /input /output '(CSE)'
```

□ Examine the output files:

Copy the output files from the distributed filesystem to the local filesystem and examine them:

```
$ bin/hdfs dfs -get output output
```

\$ cat output/* or View the output files on the distributed filesystem:

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


DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, setting up of one node Hadoop cluster was successfully executed.

EX. NO: 8**DATE:****MOUNT THE ONE NODE HADOOP CLUSTER USING FUSE****AIM:**

To mount the one node Hadoop cluster using FUSE.

PROCEDURE:

Hadoop Distributed File System (HDFS) is a distributed, scalable file system developed as the back-end storage for data-intensive Hadoop applications. As such, HDFS is designed to handle very large files with "write-once-read-many" access model. As HDFS is not a full-fledged POSIX

compliant file system, it cannot be directly mounted by the operating system, and file access with HDFS is done via HDFS shell commands.

However, one can leverage FUSE to write a userland application that exposes HDFS via a traditional file system interface. fuse-dfs is one such FUSE-based application which allows you to mount HDFS as if it were a traditional Linux file system. If you would like to mount HDFS on

Linux, you can install fuse-dfs, along with FUSE as follows.

Now, install fuse-dfs and all necessary dependencies as follows.

To install fuse-dfs on Ubuntu 12.04 and higher:

```
$ wget http://archive.cloudera.com/one-click-install/maverick/cdh3-repository_1.0_all.deb
```

```
$ sudo dpkg -i cdh3-repository_1.0_all.deb
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install hadoop-0.20-fuse
```

Once fuse-dfs is installed, go ahead and mount HDFS using FUSE as follows.

```
$ sudo hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port> <mount_point>
```

Once HDFS has been mounted at <mount_point>, you can use most of the traditional filesystem

operations (e.g., cp, rm, cat, mv, mkdir, rmdir, more, scp). However, random write operations such as rsync, and permission related operations such as chmod, chown are not supported in FUSE-mounted HDFS.

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, mounting the one node Hadoop cluster using FUSE was successfully executed.

EX. NO: 9	WRITE A PROGRAM TO USE THE API'S OF HADOOP TO INTERACT WITH IT
DATE:19.9.22	

AIM:

To write a program to use Hadoop's File system API.

PROCEDURE:

Reading data from and writing data to Hadoop Distributed File System (HDFS) can be done in a lot of ways. Now let us start by using the FileSystem API to create and write to a file in HDFS, followed by an application to read a file from HDFS and write it back to the local file system.

Step 1: Once you have downloaded a test dataset, we can write an application to read a file from the local file system and write the contents to Hadoop Distributed File System.

```
package com.hadoop.hdfs.writer;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.OutputStream;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;
import org.apache.hadoop.util.ToolRunner;
public class HdfsWriter extends Configured implements Tool {
public static final String FS_PARAM_NAME = "fs.defaultFS";
public int run(String[] args) throws Exception {
if (args.length < 2) {
System.err.println("HdfsWriter [local input path] [hdfs output path]");
return 1;
}
String localInputPath = args[0];
```

```

Path outputPath = new Path(args[1]);
Configuration conf = getConf();
System.out.println("configured filesystem = " + conf.get(FS_PARAM_NAME));
FileSystem fs = FileSystem.get(conf);
if (fs.exists(outputPath)) {
    System.err.println("output path exists");
    return 1;
}
OutputStream os = fs.create(outputPath);
InputStream is = new BufferedInputStream(new FileInputStream(localInputPath));
IOUtils.copyBytes(is, os, conf);
return 0;
}

public static void main( String[] args ) throws Exception {
    int returnCode = ToolRunner.run(new HdfsWriter(), args);
    System.exit(returnCode);
}
}

```

Step 2: Export the Jar file and run the code from terminal to write a sample file to HDFS.

```
[training@localhost ~]$ hadoop jar HdfsWriter.jar com.hadoop.hdfs.writer.HdfsWriter
sample.txt /user/training/HdfsWriter_sample.txt
```

Step 3: Verify whether the file is written into HDFS and check the contents of the file.

```
[training@localhost ~]$ hadoop fs -cat /user/training/HdfsWriter_sample.txt
```

Step 4: Next, we write an application to read the file we just created in Hadoop

Distributed File System and write its contents back to the local file system.

```

package com.hadoop.hdfs.reader;
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

```

```

import org.apache.hadoop.io.IOUtils;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class HdfsReader extends Configured implements Tool {
    public static final String FS_PARAM_NAME = "fs.defaultFS";
    public int run(String[] args) throws Exception {
        if (args.length < 2) {

            System.err.println("HdfsReader [hdfs input path] [local output path]");
            return 1;
        }
        Path inputPath = new Path(args[0]);
        String localOutputPath = args[1];
        Configuration conf = getConf();
        System.out.println("configured filesystem = " + conf.get(FS_PARAM_NAME));
        FileSystem fs = FileSystem.get(conf);
        InputStream is = fs.open(inputPath);
        OutputStream os = new BufferedOutputStream(new FileOutputStream(localOutputPath));
        IOUtils.copyBytes(is, os, conf);
        return 0;
    }

    public static void main( String[] args ) throws Exception {
        int returnCode = ToolRunner.run(new HdfsReader(), args);
        System.exit(returnCode);
    }
}

```

Step 5: Export the Jar file and run the code from terminal to write a sample file to HDFS.

```

[training@localhost ~]$ hadoop jar HdfsReader.jar com.hadoop.hdfs.reader.HdfsReader
/user/training/HdfsWriter_sample.txt /home/training/HdfsReader_sample.txt

```

Step 6: Verify whether the file is written back into local file system.

```

[training@localhost ~]$ hadoop fs -cat /user/training/HdfsWriter_sample.txt

```

FileSystem is an abstract class that represents a generic file system. Most Hadoop file system implementations can be accessed and updated through the FileSystem object. To create an instance of the HDFS, you call the method FileSystem.get(). The FileSystem.get() method will

look at the URI assigned to the fs.defaultFS parameter of the Hadoop configuration files on your

classpath and choose the correct implementation of the FileSystem class to instantiate.

The fs.defaultFS parameter of HDFS has the value hdfs://.

Once an instance of the FileSystem class has been created, the HdfsWriter class calls the create() method to create a file in HDFS. The create() method return

an OutputStream object, which can be manipulated using normal Java I/O methods.

Similarly HdfsReader calls the method open() to open a file in HDFS, which returns an InputStream object that can be used to read the contents of the file.

The FileSystem API is extensive. To demonstrate some of the other methods available in the API,

we can add some error checking to the HdfsWriter and HdfsReader classes we created.

To check whether the file exists before we call create(), use:

```
boolean exists = fs.exists(inputPath);
```

To check whether the path is a file, use:

```
boolean isFile = fs.isFile(inputPath);
```

To rename a file that already exists, use:

```
boolean renamed = fs.rename(inputPath, new Path("old_file.txt"));
```

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the program to use the Hadoop File System API to interact with it was successfully executed.

EX. NO: 10	WRITE A WORDCOUNT PROGRAM TO DEMONSTRATE THE USE OF MAP AND REDUCE TASKS
DATE:8.10.22	

AIM:

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

PROCEDURE:**STEPS:**

1. Analyze the input file content
2. Develop the code
 - a. Writing a map function
 - b. Writing a reduce function
 - c. Writing the Driver class
3. Compiling the source
4. Building the JAR file
5. Starting the DFS
6. Creating Input path in HDFS and moving the data into Input path
7. Executing the program

Sample Program:

```
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;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount
{
//Step a
```



```

public static class TokenizerMapper extends Mapper < Object , Text, Text, IntWritable >
{
//hadoop supported data types
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
//map method that performs the tokenizer job and framing the initial key value pairs
public void map( Object key, Text value, Context context) throws IOException ,
InterruptedException
{
//taking one line at a time and tokenizing the same
StringTokenizer itr = new StringTokenizer (value.toString());
//iterating through all the words available in that line and forming the key value pair
while (itr.hasMoreTokens())
{
word.set(itr.nextToken());
//sending to the context which inturn passes the same to reducer
context.write(word, one);
}
}
}
//Step b
public static class IntSumReducer extends Reducer < Text, IntWritable, Text, IntWritable >
{
private IntWritable result = new IntWritable();
// Reduce method accepts the Key Value pairs from mappers, do the aggregation based on keys
// and produce the final output
public void reduce(Text key, Iterable < IntWritable > values, Context context) throws
IOException , InterruptedException }

int sum = 0;
/*iterates through all the values available with a key and
add them together and give the final result as the key and sum of its values*/
for (IntWritable val: values)
{

```

```

sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
//Step c
public static void main( String [] args) throws Exception
{
//creating conf instance for Job Configuration
Configuration conf = new Configuration();
//Parsing the command line arguments
String [] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();
if (otherArgs.length < 2)
{
System.err.println( "Usage: wordcount <in> [<in>...]<out>" );
System.exit(2);
}
//Create a new Job creating a job object and assigning a job name for identification
//purposes
Job job = new Job(conf, "word count" );
job.setJarByClass(WordCount.class);
// Specify various job specific parameters
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
//Setting job object with the Data Type of output Key
job.setOutputKeyClass(Text.class);
//Setting job object with the Data Type of output value
job.setOutputValueClass(IntWritable.class);
//the hdfs input and output directory to be fetched from the command line
for ( int i = 0; i < otherArgs.length 1; ++i)
{
FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
}
}

```

```

}
FileOutputFormat.setOutputPath(job, new Path(otherArgs[otherArgs.length 1]));
System .exit(job.waitForCompletion( true ) ? 0 : 1);
}
}

```

Compiling the source:

```

bigdata@localhost:$ cd /home/bigdata/Downloads/mrcode/src
bigdata@localhost:/home/bigdata/Downloads/mrcode/src$ javac classpath
../lib/hadoopcommon2.5.0.jar:../lib/hadoopmapreduceclientcore2.5.0.jar:../lib/commonscli1.2.
j
ar d../build/ bigdata/WordCount.java

```

Building the JAR File:

```

bigdata@localhost:/home/bigdata/Downloads/mrcode/src$ cd ../build/
bigdata@localhost:/home/bigdata/Downloads/mrcode/build$ jar cvf wc.jar .

```

Starting the DFS (if not running already)

```

bigdata@localhost:/home/bigdata/Downloads/hadoop2.5.1$sbin/startdfs.sh

```

Creating Input path in HDFS and moving the data into Input path

```

bigdata@localhost:/home/bigdata/Downloads/hadoop2.5.1$bin/hadoop fs mkdir/mrin
bigdata@localhost:/home/bigdata/Downloads/hadoop2.5.1$bin/hadoop fs -copyFromLocal
/home/bigdata/Downloads/mrcode/mrsampleddata/* hdfs://localhost:9000/mrin

```

Executing the program

```

bigdata@localhost:/home/bigdata/Downloads/hadoop2.5.1$bin/hadoop jar
/home/bigdata/Downloads/mrcode/build/wc.jar bigdata.WordCount /mrin /mrout1

```

DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, the Word count program to use Map and reduce tasks was demonstrated successfully.

EX. NO: 11 DATE:10.10.22	FIND PROCEDURE FOR FILE TRANSFER BETWEEN VIRTUAL MACHINES
---	--

AIM

To find procedure for file transfer between virtual machines.

PROCEDURE:

Step 1: Start the Virtualbox Guest Machine (OS).

Step 2: From Oracle's VM VirtualBox main menu, select Devices Install Guest Additions

Step 3: Open Windows Explorer.

Step 4: Double click at the "CD Drive (X) VirtualBox Guest additions to explore its

contents Step 5: Right click at "VBoxWindowsAdditions application and from the pop-up menu, choose "Run as administrator"

Step 6: Press Next and then follow the on screen instructions to complete the Guest Additions installation

Step 7: When the setup is completed, choose Finish and restart the Virtualbox guest machine.

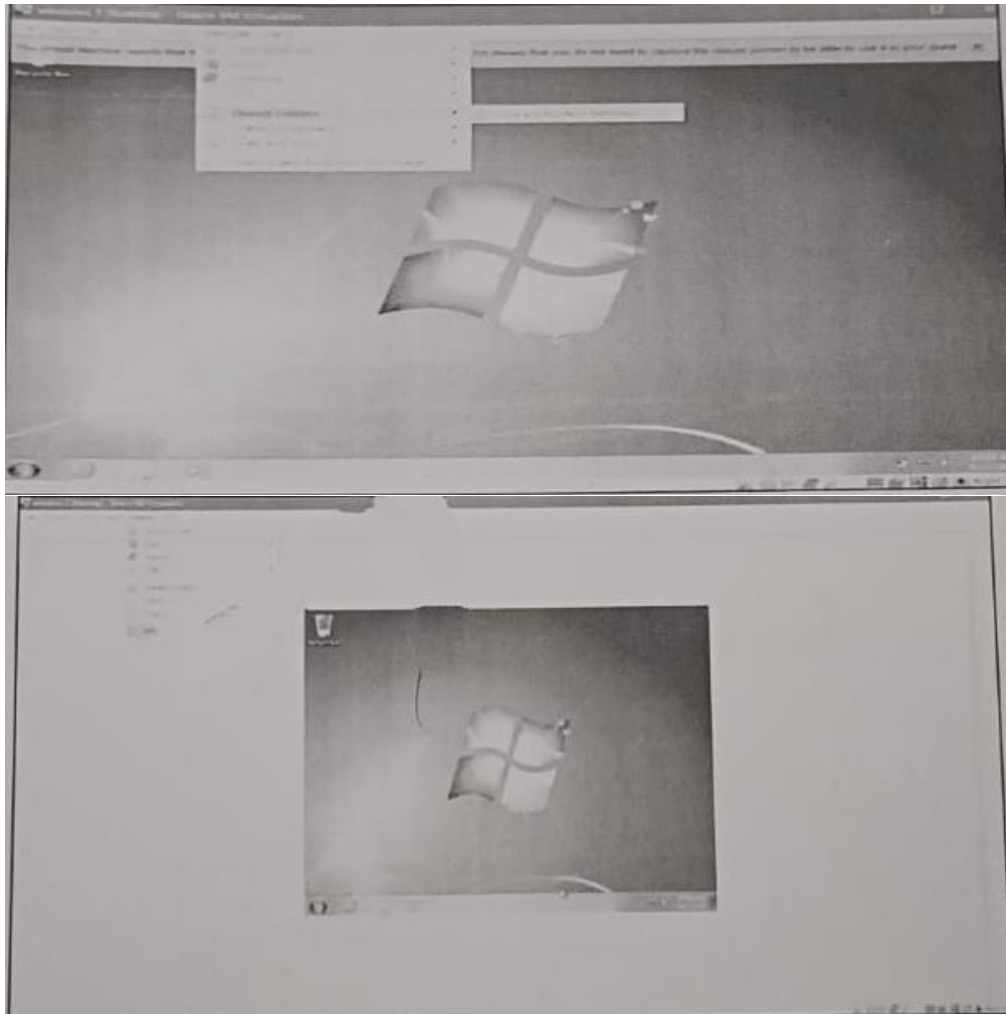
Step 8: From the VirtualBox menu click Devices and choose Shared Folders →→ Shared Folder Settings

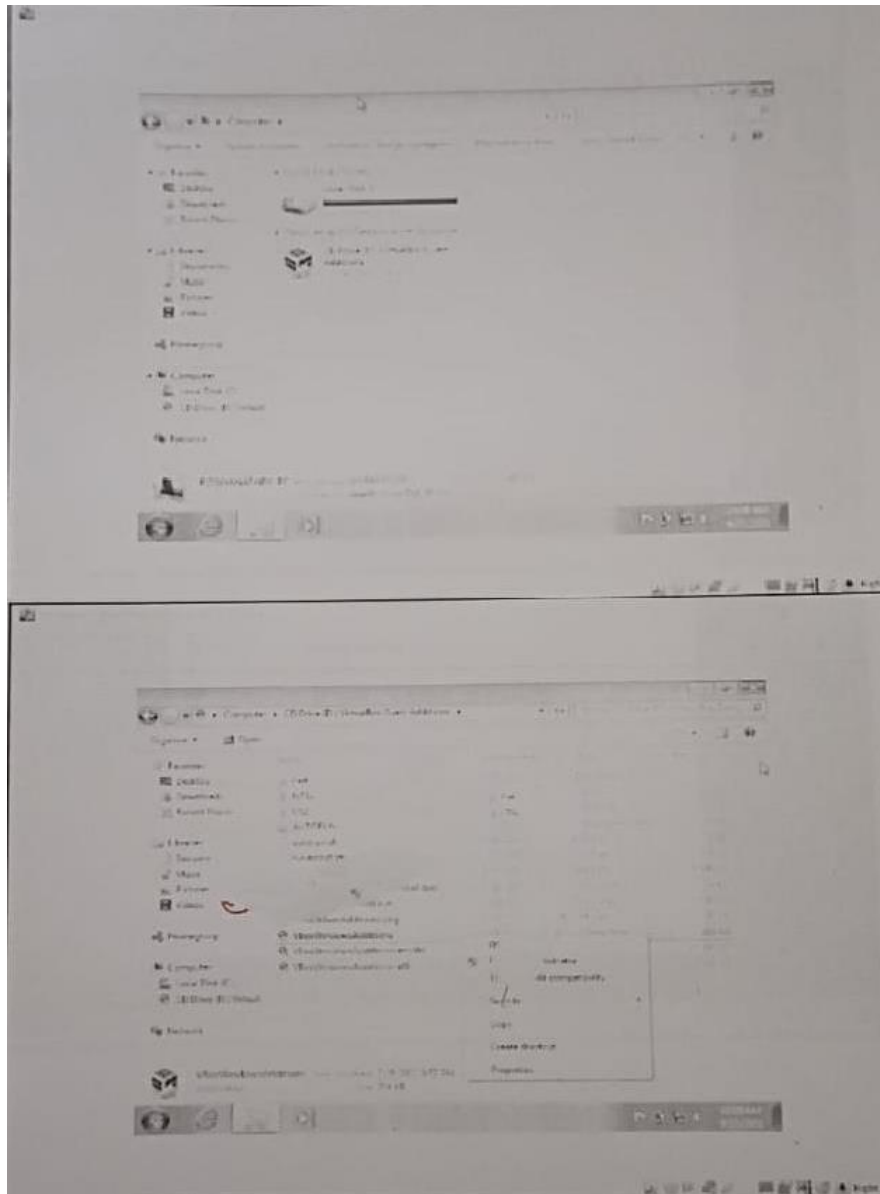
Step 9: Click the Add new shared folder icon. Create a new folder for the file sharing on the Host OS and give it a recognizable name. (e.g. "Public")

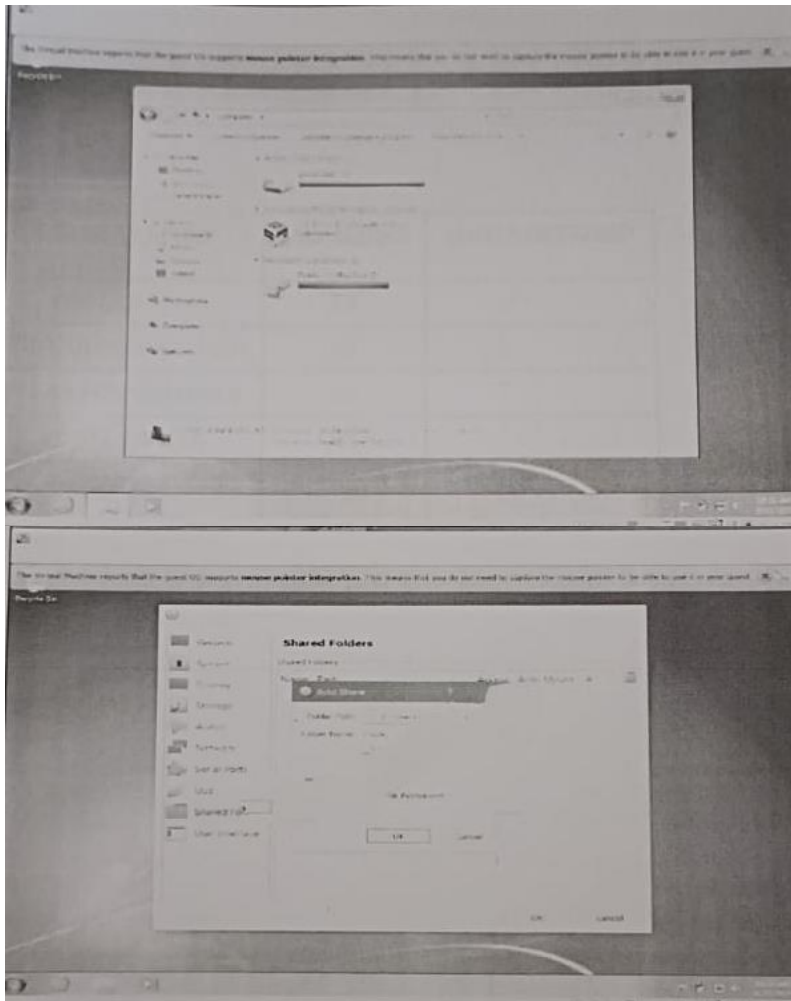
Step 10: Now, in the 'Add Share' options, type a name at the 'Folder Name box, click the AutoMount and the Make Permanent checkboxes and click OK twice to close the Shared Folder Settings.

Step 11: To access the shared folder from the Guest OS, open Windows Explorer and under the Network locations' you should see a new network drive that corresponds to the shared folder on the Host OS.

OUTPUT:







DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, to find procedure to transfer files from one machine to another has been completed successfully

EX. NO: 12
DATE:17.10.22

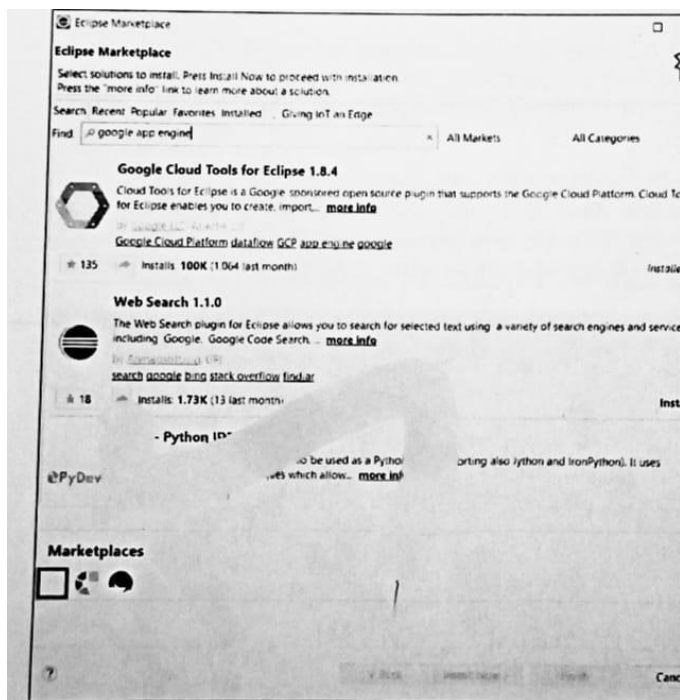
HOSTING STATIC WEBPAGE IN GOOGLE APP ENGINE

AIM:

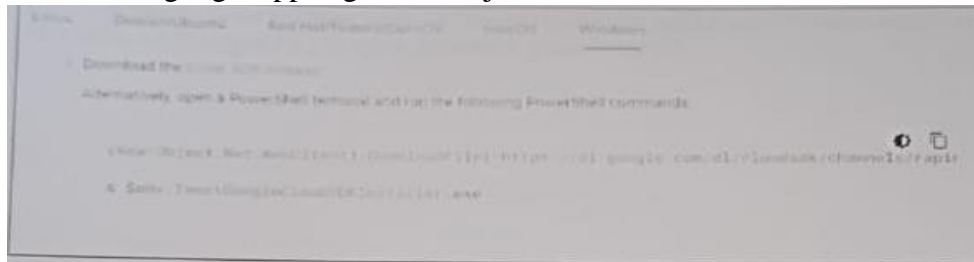
To create a java web application using google app engine.

PROCEDURE:

1. Install Eclipse IDE and Java JDK
2. In Eclipse IDE, go to Windows > Eclipse Marketplace > Google Cloud Tools for Eclipse 1.8.4>> Click install. The google app engine plugin will be installed.



1.Install the google app engine cloud java SDK installer

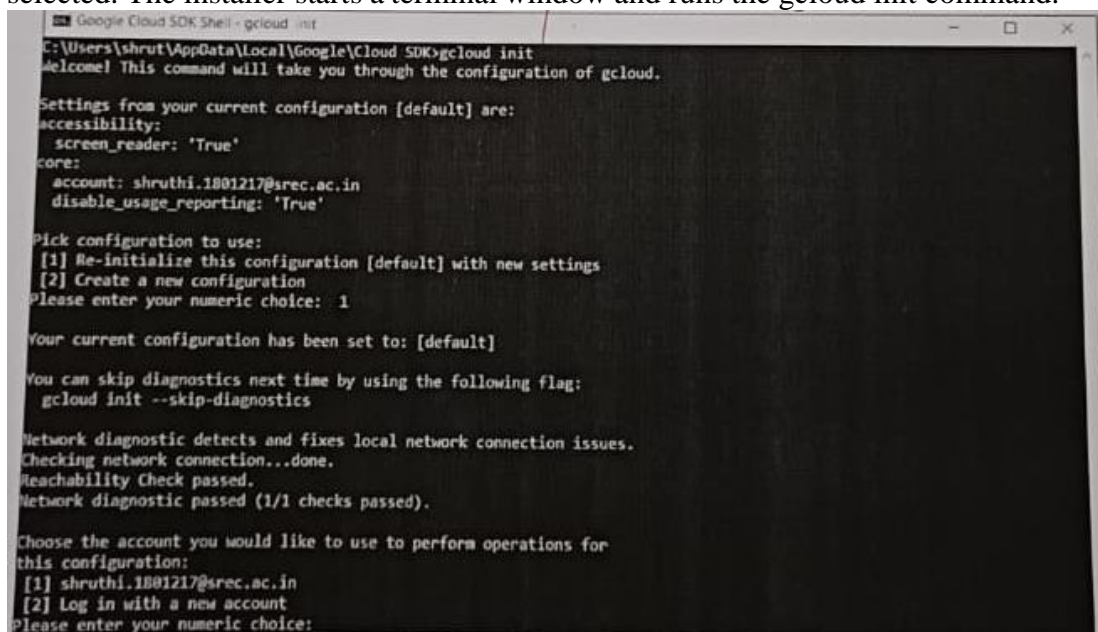


1.Launch the installer. The installer is signed by Google LLC.

1. Cloud SDK requires Python, supported versions are Python 3 (preferred, 3.5 to 3.8) and Python 2 (2.7.9 or higher). Cloud SDK comes bundled with Python 3 by

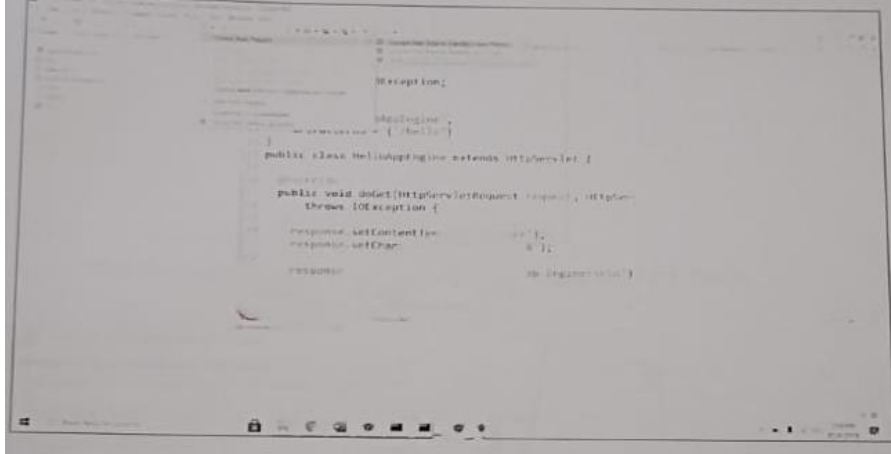
default. To use Cloud SDK, your operating system must be able to run a supported version of Python.

1. After installation is complete, the installer gives you the option to create Start Menu and Desktop shortcuts, start Cloud SDK shell, and configure the Cloud SDK. Make sure that you leave the options to start the shell and configure your installation selected. The installer starts a terminal window and runs the `gcloud init` command.

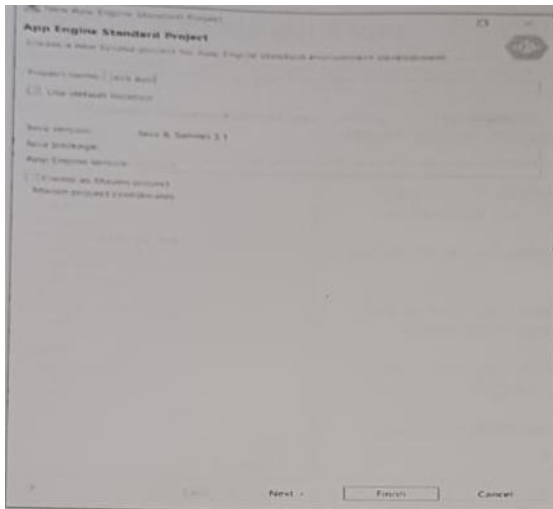


1 Install the java cloud component using the command `gcloud install component app-engine-java` 1. Create New Web Application Project

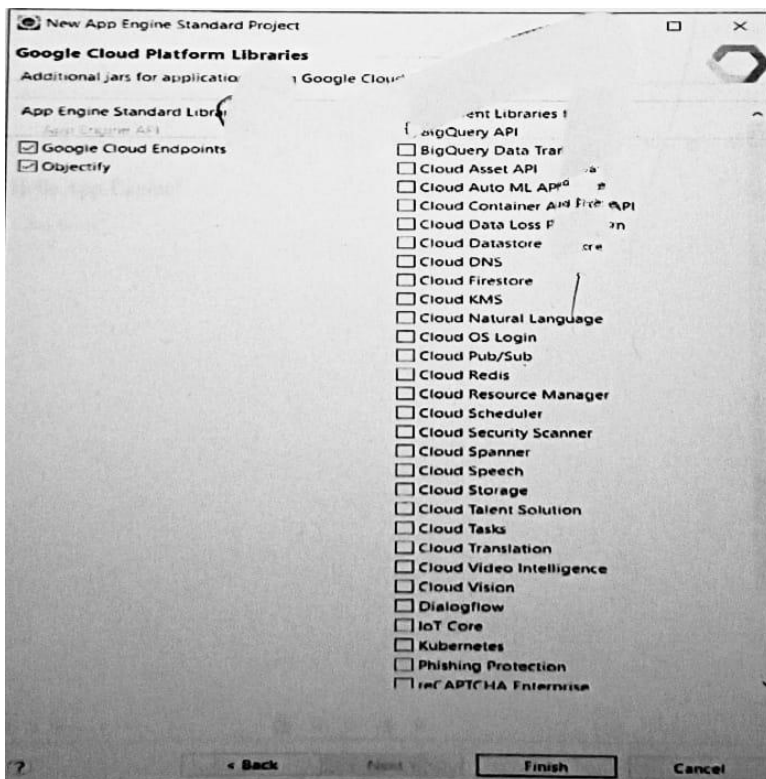
1. In the Eclipse toolbar, click on the Google icon, and select "Create New Project" and click on "Google App Engine Standard Java Project".



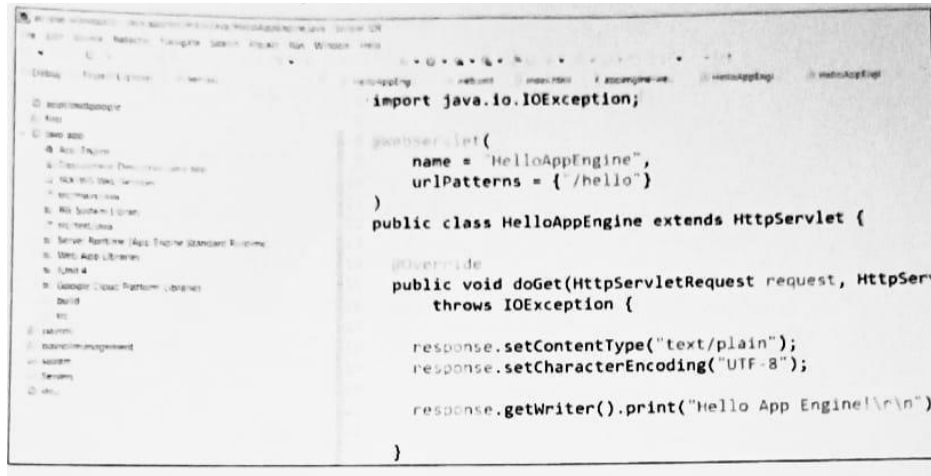
1. Give a name for the application and click next



1. Click the App Engine API and objectify check boxes and click Finish.



Review the generated project directory



1 Run it locally. Right click on the project and run on Web Server. The project will run in the localhost



DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, static web page is hosted in google app engine successfully

EX. NO: 13**OPENNEBULLA INSTALLATION****DATE:3.10.22****AIM:**

To install opennebula and to run it.

PROCEDURE:

Step 1: Add OpenNebula and Debian repositories,

Run the following commands to add epel and OpenNebula repositories on Ubuntu.

```
wget -q -O- https://downloads.opennebula.org/repo/repo.key | sudo apt-key add
```

Step 2: Add the repository to the system using the following command

```
echo "deb https://downloads.opennebula.org/repo/6.1/Ubuntu/20.04 opennebula" | sudo tee /etc/apt/sources.list.d/opennebula.list
```

```
stable
```

Step 3: Install OpenNebula Front-end packages These are the packages available on OpenNebula CentOS repository.

```
sudo apt update
```

```
sudo apt install opennebula opennebula-sunstone opennebula-gate opennebula-flow
```

Step 4: Ruby Runtime Installation. `sudo /usr/share/one/install_gems`

Step 5: Configure oneadmin credentials

```
$ sudo su oneadmin
```

```
$ echo "oneadmin.mypassword"> ~/.one/one_auth
```

Step 5: Start OpenNebula daemons.

```
sudo systemctl start opennebula opennebula-sunstone sudo systemctl enable opennebula opennebula-sunstone
```

Step 6: Display the configuration details

Step 7: Access open nebula sunstone in localhost <http://localhost.9869>

The image shows two screenshots of a terminal window on a virtual machine named 'rashan-VirtualBox'. The date is Dec 9, 2021.

Top Screenshot: The terminal shows the installation of Ruby 2.7.0 and its dependencies. The user is at the root prompt. The output shows the following steps:

```

Preparing to unpack .../2-ruby2.7_2.7.0-Subuntui.5_and64.deb ...
Unpacking ruby2.7 (2.7.0-Subuntui.5) ...
Selecting previously unselected package ruby.
Preparing to unpack .../3-ruby_1:3.0.7-1_amd64.deb ...
Unpacking ruby (1:3.0.7-1) ...
Selecting previously unselected package javascript-common.
Preparing to unpack .../4-javascript-common_11_all.deb ...
Unpacking javascript-common (11) ...
Selecting previously unselected package libjs-jquery.
Preparing to unpack .../5-libjs-jquery_3.3.1-dfsg-3_all.deb ...
Unpacking libjs-jquery (3.3.1-dfsg-3) ...
Setting up javascript-common (11) ...
Setting up libjs-jquery (3.3.1-dfsg-3) ...
Setting up ruby-xmlrpc (0.3.0-2) ...
Setting up rake (13.0.1-2) ...
Setting up libruby2.7:amd64 (2.7.0-Subuntui.5) ...
Setting up ruby2.7 (2.7.0-Subuntui.5) ...
Setting up ruby (1:3.0.7-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...
root@rashan-VirtualBox:~# cd /usr/share/zone/install_gems
-bash: /usr/share/zone/install_gems: no such file or directory
root@rashan-VirtualBox:~# cd /usr/share/zone/install_gems
-bash: /usr/share/zone/install_gems: no such file or directory
root@rashan-VirtualBox:~# cd /usr/share/zone/install_gems
-bash: /usr/share/zone/install_gems: no such file or directory
root@rashan-VirtualBox:~#

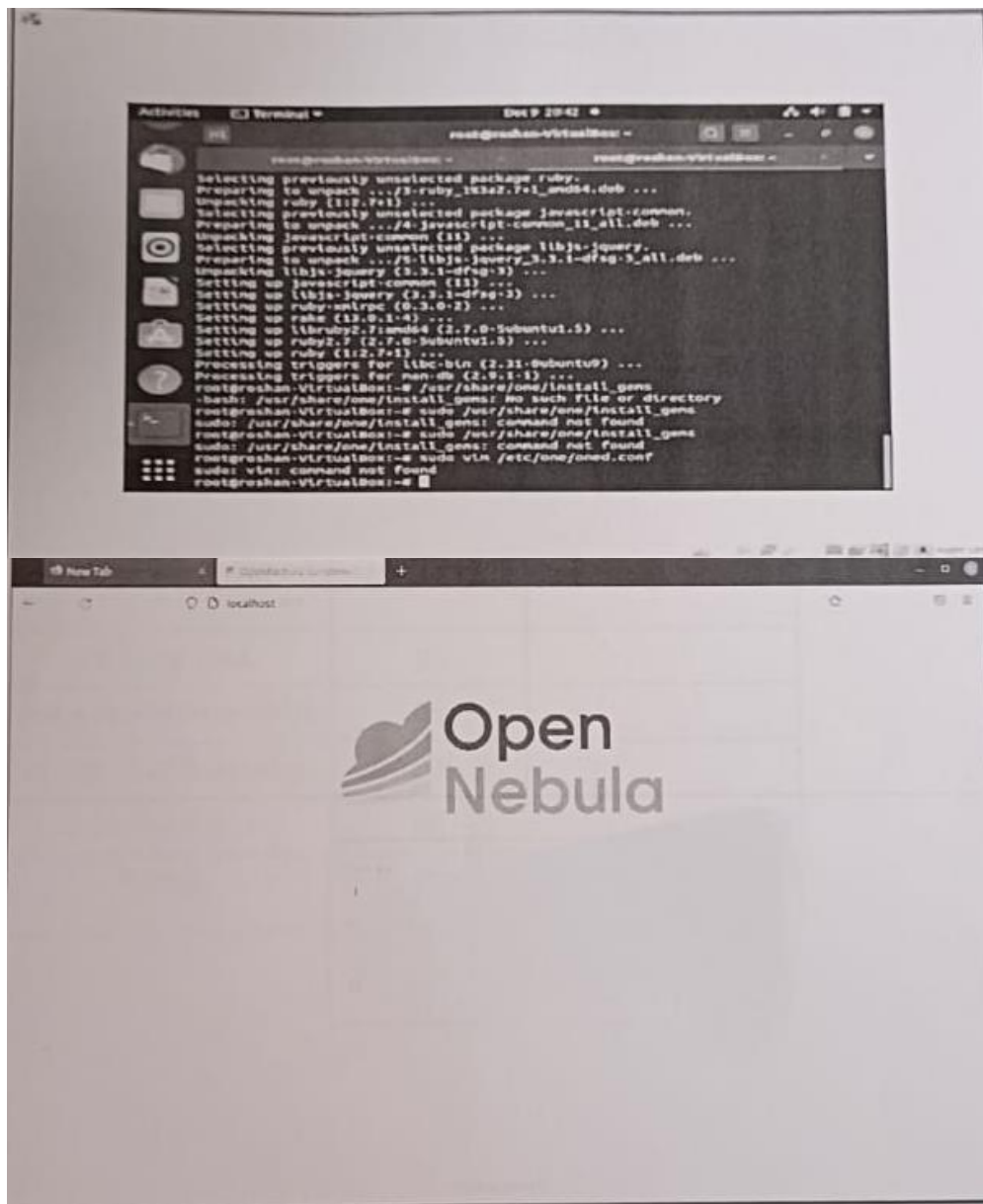
```

Bottom Screenshot: The terminal shows the output of the command `apt --fix-broken install`. The output indicates that the following packages were automatically installed and are no longer required:

```

funky-tako javascript-common libjs-jquery libruby2.7 openssl-common
openssl-core-common openssl-ruby ruby ruby-minitest ruby-net-telnet
ruby-power-assert ruby-test-unit ruby-xmlrpc ruby2.7 rubygems-integration
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  javascript-common libjs-jquery libruby2.7 ruby ruby-xmlrpc ruby2.7
Suggested packages:
  apache2 | lighttpd | httpd ri ruby-dev
The following NEW packages will be installed:
  javascript-common libjs-jquery libruby2.7 ruby ruby2.7
The following packages will be upgraded:
  ruby-xmlrpc
1 upgraded, 2 newly installed, 0 to remove and 486 not upgraded.
Need to get 0 B/3.986 kB of archives.
After this operation, 18.4 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
(Reading database ... 188964 files and directories currently installed.)
Preparing to unpack .../0-ruby-xmlrpc_0.3.0-2_all.deb ...
Unpacking ruby-xmlrpc (0.3.0-2) over (0.3.0-2) ...

```





DESCRIPTION	MARKS
PRE-VIVA(5)	
PRE-LAB PREPARATION(6)	
IN LAB PERFORMANCE(7)	
POST LAB(2)	
TOTAL(20)	

RESULT:

Thus, open nebula installation has been completed successfully.

