How to setup OpenStack based cloud computing platform for IoT (Internet of Things) motion control applications

Author: Jing Yan ([jing.yan@aalto.fi](mailto:jing.yan@aalto.fi))

Updated: May 23rd, 2022

[1. How to setup Framework 1 (Host) on APC 910 2](#_Toc104895775)

[1.1 Introduction of Framework 1 2](#_Toc104895776)

[1.2 Steps to setup Framework 1 3](#_Toc104895777)

[2. How to setup Framework 2 (qemu-kvm based VM) on APC 910 4](#_Toc104895778)

[2.1 Introduction of Framework 2 4](#_Toc104895779)

[2.2 Steps to setup Framework 2 4](#_Toc104895780)

[3. How to setup Framework 3 (OpenStack) on APC 910 7](#_Toc104895781)

[3.1 Introduction of Framework 3 7](#_Toc104895782)

[3.1.1 Main services in OpenStack 7](#_Toc104895783)

[3.1.2 How Nova Service works and interacts with other services in OpenStack 8](#_Toc104895784)

[3.1.3 Architecture of Neutron service in OpenStack 8](#_Toc104895785)

[3.1.4 Comparison between “OpenvSwitch” and “Linux Bridge” provided by Neutron service in OpenStack 9](#_Toc104895786)

[3.1.4.1 Advantages of OpenvSwitch 9](#_Toc104895787)

[3.1.4.2 Disadvantages of OpenvSwitch 10](#_Toc104895788)

[3.1.4.3 Advantages of Linux Bridge 10](#_Toc104895789)

[3.1.4.4 Disadvantages of Linux Bridge 10](#_Toc104895790)

[3.1.5 Mechanism of OpenvSwitch with Flat provided by Neutron service 11](#_Toc104895791)

[3.2 Steps to setup Framework 3 11](#_Toc104895792)

[3.2.1 Keystone service (Controller Node) 12](#_Toc104895793)

[3.2.2 Glance service (Controller Node) 14](#_Toc104895794)

[3.2.3 Nova service (Controller Node) 16](#_Toc104895795)

[3.2.4 Nova service (Computing Node) 20](#_Toc104895796)

[3.2.5 Neutron service (Controller Node) 22](#_Toc104895797)

[3.2.6 Neutron service (Computing Node) 27](#_Toc104895798)

[3.2.7 Security group and flavour configuration (Controller Node) 30](#_Toc104895799)

[3.2.8 Horizon service (Controller Node) 31](#_Toc104895800)

[3.2.9 Start the instance through web UI 32](#_Toc104895801)

[4. How to setup Framework 4 (Kubernetes) on APC 910 34](#_Toc104895802)

[4.1 Introduction of Framework 4 34](#_Toc104895803)

[4.2 Steps to setup Framework 4 34](#_Toc104895804)

[4.2.1 How to deploy Kubernetes 35](#_Toc104895805)

[4.2.2 How to make a Docker image for CODESYS Control 37](#_Toc104895806)

[5. Connection between the frameworks and the IoT motion control target 38](#_Toc104895807)

[5.1 Introduction of the connection between the frameworks and IoT motion control target 38](#_Toc104895808)

[5.2 CODESYS Program to test the connection between frameworks and the IoT motion control target 39](#_Toc104895809)

[5.3 How to Create Windows 10 Image for OpenStack VM 39](#_Toc104895810)

[5.3.1 Introduction about the background 39](#_Toc104895811)

[5.3.2 Steps to make Window 10 Image for OpenStack VM 39](#_Toc104895812)

[6. How to login to our OpenStack platform 62](#_Toc104895813)

# 1. How to setup Framework 1 (Host) on APC 910

## 1.1 Introduction of Framework 1

The following figure shows the architecture of Framework 1, it means the APC 910 with Debian 11 OS (Operation System) with real-time kernel running on it.

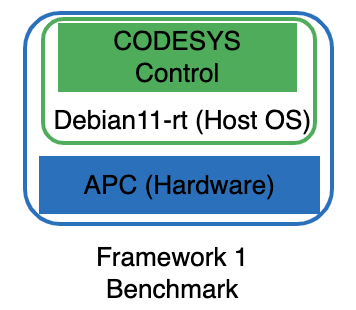


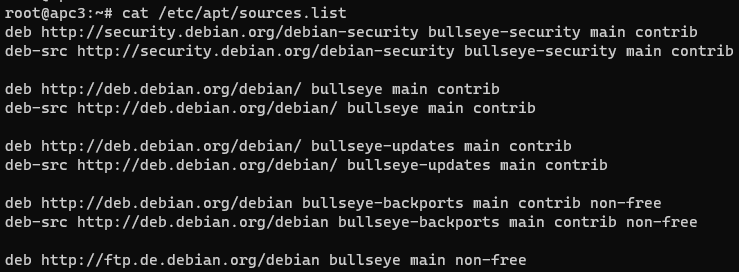
Figure 1: Architecture of framework 1

## 1.2 Steps to setup Framework 1

1. Download the ISO file from this [link](https://cdimage.debian.org/debian-cd/current/amd64/iso-dvd/debian-11.3.0-amd64-DVD-1.iso), burn the ISO file to a USB disk and install the OS on the host. For example, in my case, I was using a Debian 11 ISO file, and running the “rufus” tool on Windows 10 to burn the Debian 11 ISO file to a USB disk.

2. Install the Debian 11 to APC 910 with the USB bootable disk which was made in the previous step based on your own requirements, for example, with “ssh service”, without “Desktop”.

3. Start the Debian 11 OS on APC 910, configure repos for this machine in “/etc/apt/sources.list” as follows, then run “apt update”



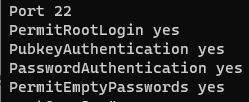
4. Install the real-time kernel

For example, if you have the following first kernel package installed, you need to replace the original one with the second one with “rt.”

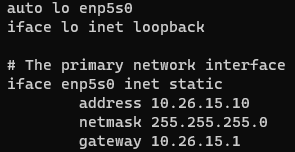
<https://packages.debian.org/stable/kernel/linux-image-5.10.0-10-amd64>

<https://packages.debian.org/stable/kernel/linux-image-5.10.0-10-rt-arm64>

5. Configure ssh service based on your own requirements, for example: uncomment the following configuration in “/etc/ssh/sshd\_config”, then restart ssh service by “systemctl restart sshd”, now you can use “ssh username@ip” to access this machine remotely.



6. Configure the IP (Internet Protocol) address for Ethernet based on your own requirements, for example, I configured a static IP for the Ethernet interface in “/etc/network/interfaces”, then restart networking service by “systemctl restart networking”, now this machine can be accessed by the static IP I set.



# 2. How to setup Framework 2 (qemu-kvm based VM) on APC 910

## 2.1 Introduction of Framework 2

The following figure shows the architecture of Framework 2, it means the Virtual Machine (VM) with real-time kernel is running on the Host with Debian 11 OS real-time kernel which is running on APC 910.

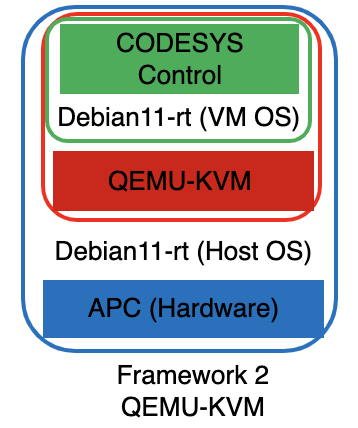
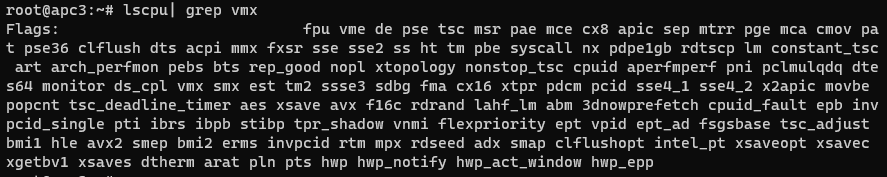


Figure 2: Architecture of framework 2

## 2.2 Steps to setup Framework 2

1. Finish the “How to setup Framework 1 (Host) on APC 910”, now we are going to install virtualization components on this physical machine.

2. Verify the virtualization function is enabled in BIOS through the following command, if there is no output for the following command, you must enable the VT technology in BIOS.

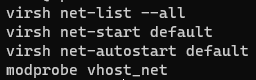


3. Install necessary packages



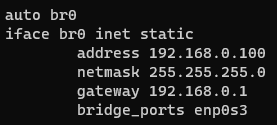
4. Configure network for VM

4.1 Configure the virtual network if you are going to use NAT (Network Address Translation) mode for VM’s network configuration by executing the following commands.

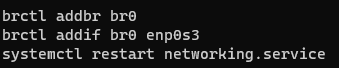


4.2 Configure a bridge network for the host if you are going to make the VM share the same Ethernet interface with the host through the bridge by executing the following command (Recommended).

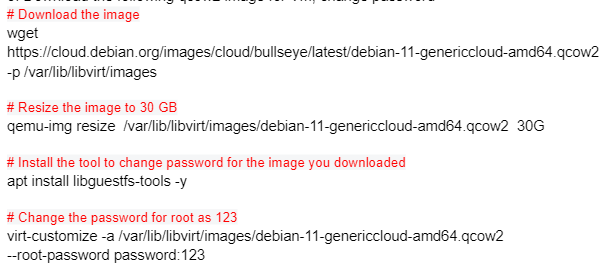
First, reconfigure the “/etc/network/interfaces” file on host based on your own needs, for example:



Second, execute the following commands.



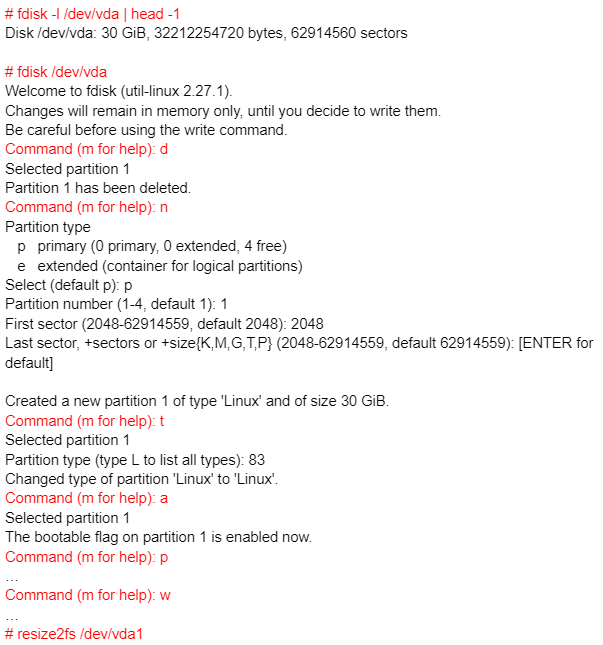
5. Download the following qcow2 image for VM, change password



6. Install the VM and set the bridge network for the VM

Based on the “br0” created above and the image downloaded above, start the virt-manager program, and install the VM with the GUI (Graphical User Interface) tool “virt-manager”. (If you do not know how to use virt-manager, pls google it, it is quite easy.)

7. Start the VM and login to the VM to expand the storage for VM so that the VM can take up the entire disk.



8. Install real-time kernel for the VM, enable ssh and configure the IP address as what you have done for the Framework 1.

# 3. How to setup Framework 3 (OpenStack) on APC 910

## 3.1 Introduction of Framework 3

The following figure shows the architecture of Framework3, if you are not familiar with OpenStack, pls google it, there are lots of open source materials you could access in the Internet.

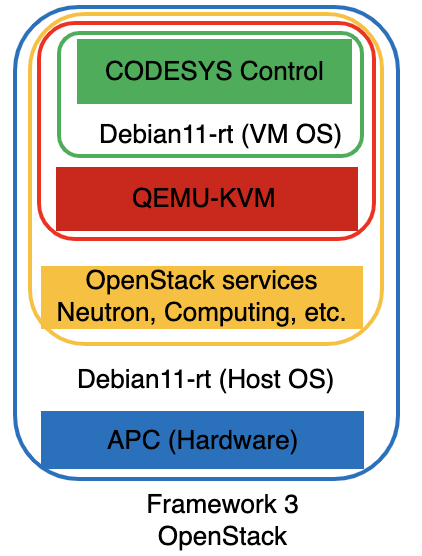


Figure 3: Architecture of framework 3

### 3.1.1 Main services in OpenStack

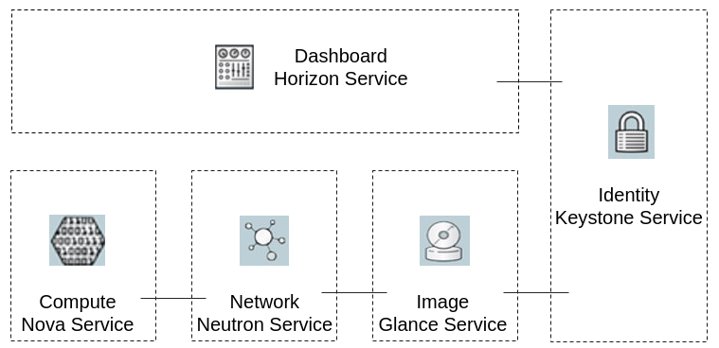


Figure 4: Main services of OpenStack

### 3.1.2 How Nova Service works and interacts with other services in OpenStack

Here is the more detailed information about each sub-service

* Compute: manages communication with hypervisor and virtual machines.​
* Scheduler: decides which host gets each instance​
* Conductor: handles requests that need coordination(build/resize), acts as a database proxy, or handles object conversions.​
* API (Application Programming Interface): component that receives HTTP (Hypertext Transfer Protocol) requests, converts commands and communicates with other components via the oslo.messaging queue or HTTP

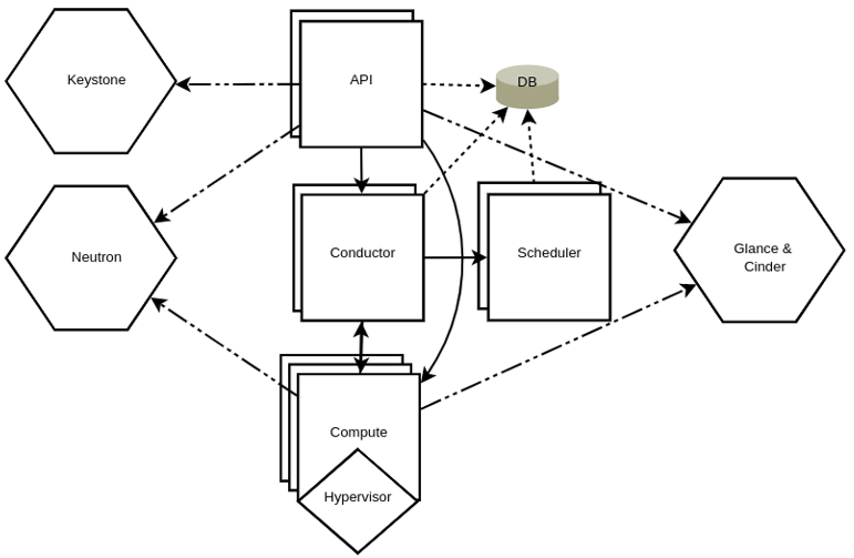


Figure 5: How Nova works and interacts with other services in OpenStack

### 3.1.3 Architecture of Neutron service in OpenStack

There are two kinds of “type drivers” provided by OpenStack, namely “Linux Bridge” and “OpenvSwitch,” which are managed by the “ML2” plugin. After configuring the “type drivers” through the “ML2” plugin, you can configure the “mechanism drivers” for the “type drivers”, including “local”, “flat”, “VLAN”, “VXLAN” and “GRE”.

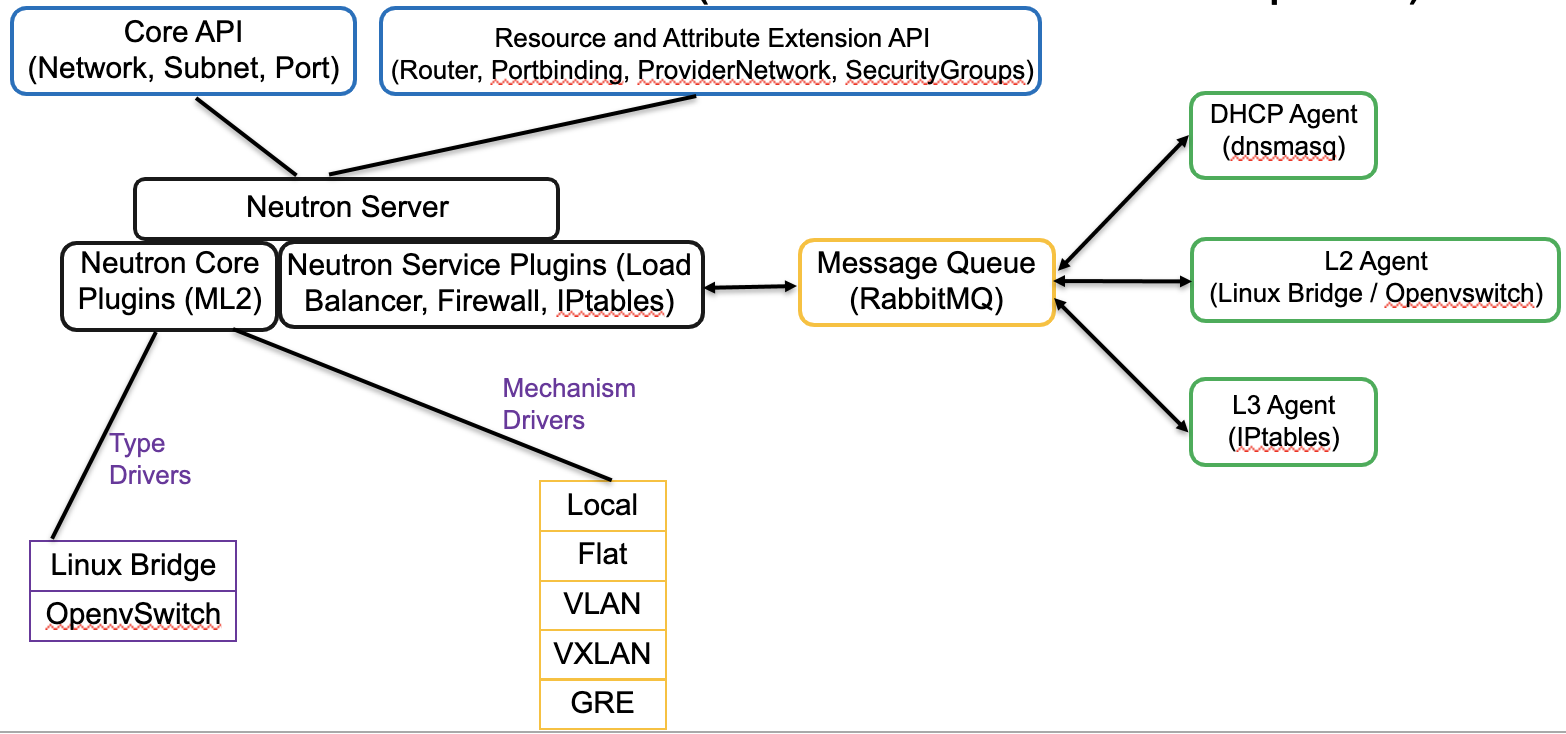


Figure 6: How Neutron service works in OpenStack

### 3.1.4 Comparison between “OpenvSwitch” and “Linux Bridge” provided by Neutron service in OpenStack

Based on the comparison above, I finally chose “OpenvSwitch” with “Flat” in my thesis and I will explain more about the mechanism of “OpenvSwitch” with “Flat.”

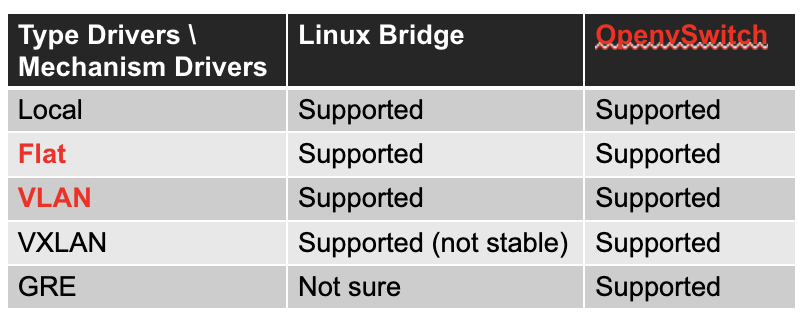


Figure 7: Comparison between “OpenvSwitch” and “Linux Bridge” provided by Neutron service in OpenStack

#### 3.1.4.1 Advantages of OpenvSwitch

* Easier for network management – With the OpenvSwitch, it is convenient for the administrator to manage and monitor the network status and data flow in the cloud environment.​
* Support more tunnel protocols – OVS supports GRE, VXLAN, IPsec, etc.
* Incorporated in SDN (Software Defined Networking) – OpenvSwitch is incorporated in software-defined networking (SDN) that can be driven by using an OpenStack plug-in or directly from an SDN Controller, such as OpenDaylight.

#### 3.1.4.2 Disadvantages of OpenvSwitch

* Lacks stability – OpenvSwitch has some stability problems such as Kernel panics, ovs-switched segfaults, and data corruption.​
* Complex operation – OpenvSwitch itself is a complex solution, which owns so many functions. It is hard to learn, install and operate.

#### 3.1.4.3 Advantages of Linux Bridge

* Stable and reliable – Linux Bridge has been used for years, its stability and reliability are approved.​
* Easy for installation – Linux Bridge is a part of standard Linux installation and there are no additional packages to install or learn.​
* Convenient for troubleshooting – Linux Bridge itself is a simple solution that its operation is simpler than that of Open vSwitch. It is convenient for troubleshooting.

#### 3.1.4.4 Disadvantages of Linux Bridge

* Fewer functions – Linux Bridge does not support the Neutron DVR.​
* Fewer supporters – Many enterprises wanted to ensure that there was an open model for integrating their services into OpenStack. However, Linux Bridge cannot ensure the demand, so it has fewer users than that of OpenvSwitch.

### 3.1.5 Mechanism of OpenvSwitch with Flat provided by Neutron service

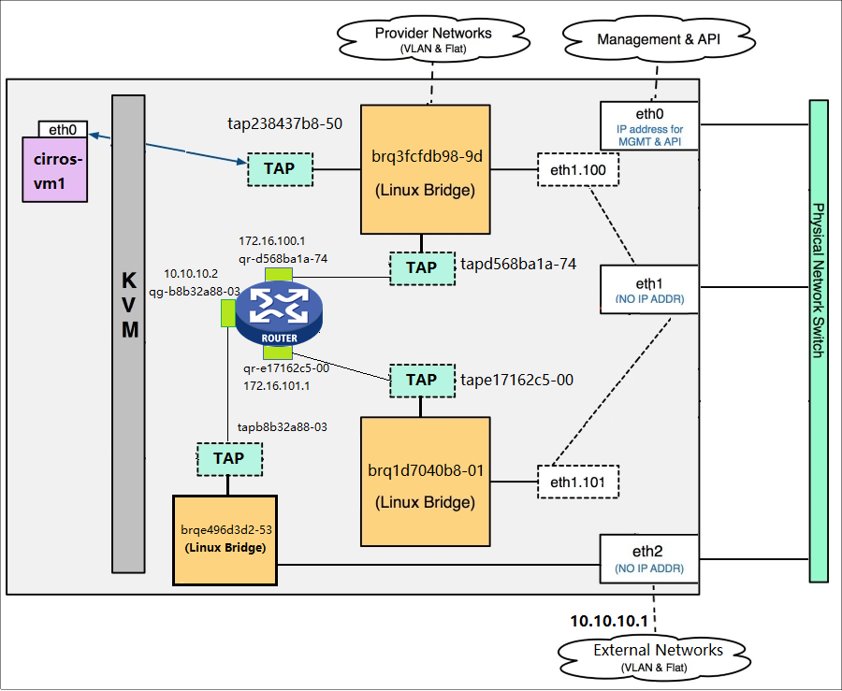


Figure 8: Mechanism of OpenvSwitch with Flat provided by Neutron service

## 3.2 Steps to setup Framework 3

Now, I assume you have two APC 910 with Debian 11 - real-time kernel installed.

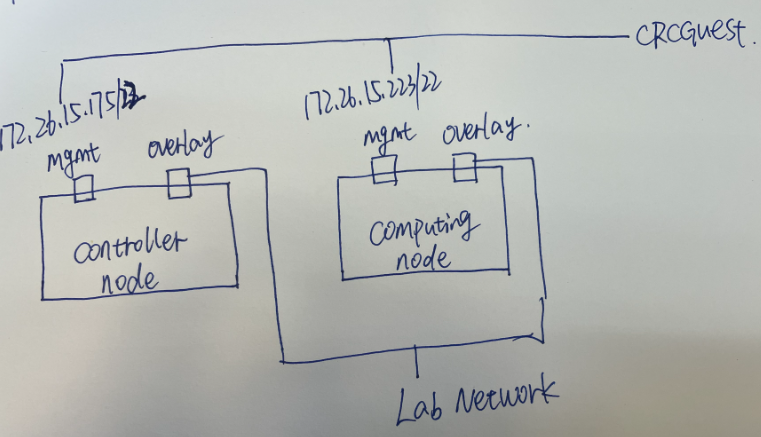


Figure 9: Rough architecture of Openstack deployment

### 3.2.1 Keystone service (Controller Node)

Note, this service is deployed on the first node, which also acts as the controller node.

1. Install the backend database packages, configure mysql, add rabbitmq user and set permissions. Modify mariadb, memcached configuration files and start/enable services.

apt -y install mariadb-server rabbitmq-server memcached python3-pymysql

systemctl enable mariadb; systemctl restart mariadb

---------------------------------------------------

mysql\_secure\_installation

Enter current password for root (enter for none):

Switch to unix\_socket authentication [Y/n] n

Change the root password? [Y/n] n

Remove anonymous users? [Y/n] y

Disallow root login remotely? [Y/n] y

Remove test database and access to it? [Y/n] y

Reload privilege tables now? [Y/n] y

---------------------------------------------------

rabbitmqctl add\_user openstack password

rabbitmqctl set\_permissions openstack ".\*" ".\*" ".\*"

---------------------------------------------------

root@dlp:~# vi /etc/mysql/mariadb.conf.d/50-server.cnf

# line 30 : change

bind-address = 0.0.0.0

# line 43 : uncomment and change

# default value 151 is not enough on Openstack Env

max\_connections = 500

---------------------------------------------------

---------------------------------------------------

root@dlp:~# vi /etc/memcached.conf

# line 35 : change

-l 0.0.0.0

---------------------------------------------------

systemctl restart mariadb rabbitmq-server memcached

systemctl enable rabbitmq-server memcached

2. Configure keystone permission in mysql database, install keystone package, modify the configuration files and start/enable apache service.

---------------------------------------------------

root@dlp:~# mysql

MariaDB [(none)]> create database keystone; grant all privileges on keystone.\* to keystone@'localhost' identified by 'password'; grant all privileges on keystone.\* to keystone@'%' identified by 'password'; flush privileges;

MariaDB [(none)]> exit

Bye

---------------------------------------------------

apt -y install keystone python3-openstackclient apache2 libapache2-mod-wsgi-py3 python3-oauth2client

---------------------------------------------------

root@dlp:~# vi /etc/keystone/keystone.conf

# line 360 : add Memcache Server info, replace the IP address based on your own environment

memcache\_servers = 172.26.15.175:11211

# line 506 : add MariaDB connection info

connection = mysql+pymysql://keystone:password@172.26.15.175/keystone

# line 2069 : uncomment

provider = fernet

---------------------------------------------------

su -s /bin/bash keystone -c "keystone-manage db\_sync"

keystone-manage fernet\_setup --keystone-user keystone --keystone-group keystone

keystone-manage credential\_setup --keystone-user keystone --keystone-group keystone

keystone-manage bootstrap --bootstrap-password adminpassword --bootstrap-admin-url http://172.26.15.175:5000/v3/ --bootstrap-internal-url http://172.26.15.175:5000/v3/ --bootstrap-public-url http://172.26.15.175:5000/v3/ --bootstrap-region-id RegionOne

echo "ServerName 172.26.15.175" >> /etc/apache2/apache2.conf

systemctl restart apache2; systemctl enable apache2

3. Configure the credential file for keystone to access openstack

root@dlp:~# vi ~/keystonerc

export OS\_PROJECT\_DOMAIN\_NAME=default

export OS\_USER\_DOMAIN\_NAME=default

export OS\_PROJECT\_NAME=admin

export OS\_USERNAME=admin

export OS\_PASSWORD=adminpassword

export OS\_AUTH\_URL=http://172.26.15.175:5000/v3

export OS\_IDENTITY\_API\_VERSION=3

export OS\_IMAGE\_API\_VERSION=2

export PS1='\u@\h \W(keystone)\$ '

---------------------------------------------------

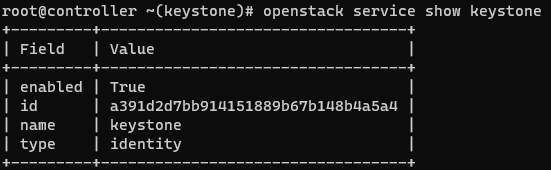
chmod 600 ~/keystonerc; source ~/keystonerc

root@dlp ~(keystone)# echo "source ~/keystonerc " >> ~/.bashrc

openstack project create --domain default --description "Service Project" service

openstack project list

4. After installation, you can see keystone service in openstack controller node



### 3.2.2 Glance service (Controller Node)

1. Configure Openstack and mysql user/permission for glance service

root@dlp ~(keystone)#

openstack user create --domain default --project service --password servicepassword glance

openstack role add --project service --user glance admin

openstack service create --name glance --description "OpenStack Image service" image

openstack endpoint create --region RegionOne image public http://172.26.15.175:9292

openstack endpoint create --region RegionOne image internal http://172.26.15.175:9292

openstack endpoint create --region RegionOne image admin http://172.26.15.175:9292

root@dlp ~(keystone)# mysql

MariaDB [(none)]> create database glance; grant all privileges on glance.\* to glance@'localhost' identified by 'password'; grant all privileges on glance.\* to glance@'%' identified by 'password'; flush privileges;

MariaDB [(none)]> exit

Bye

2. Install the package and modify the configuration files

apt -y install glance

mv /etc/glance/glance-api.conf /etc/glance/glance-api.conf.bak

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/glance/glance-api.conf

# create new

[DEFAULT]

bind\_host = 0.0.0.0

[glance\_store]

stores = file,http

default\_store = file

filesystem\_store\_datadir = /var/lib/glance/images/

[database]

# MariaDB connection info

connection = mysql+pymysql://glance:password@172.26.15.175/glance

# keystone auth info

[keystone\_authtoken]

www\_authenticate\_uri = http://172.26.15.175:5000

auth\_url = http://172.26.15.175:5000

memcached\_servers = 172.26.15.175:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = glance

password = servicepassword

[paste\_deploy]

flavor = keystone

3. Enable/Start glance service and upload the image to openstack through glance service

chmod 640 /etc/glance/glance-api.conf

chown root:glance /etc/glance/glance-api.conf

su -s /bin/bash glance -c "glance-manage db\_sync"

systemctl restart glance-api

systemctl enable glance-api

wget https://cdimage.debian.org/cdimage/openstack/current-10/debian-10-openstack-amd64.qcow2 -P /var/kvm/images

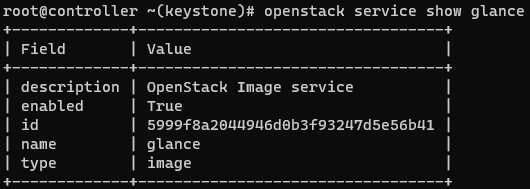
apt install libguestfs-tools -y

virt-customize -a /var/kvm/images/debian-10-openstack-amd64.qcow2 --root-password password:123

openstack image create "Debian10" --file /var/kvm/images/debian-10-openstack-amd64.qcow2 --disk-format qcow2 --public

openstack image list

4. After installation, you can see glance service in openstack controller node



### 3.2.3 Nova service (Controller Node)

1. Configure Openstack and mysql user/permission for nova service

openstack user create --domain default --project service --password servicepassword nova

openstack role add --project service --user nova admin

openstack user create --domain default --project service --password servicepassword placement

openstack role add --project service --user placement admin

openstack service create --name nova --description "OpenStack Compute service" compute

openstack service create --name placement --description "OpenStack Compute Placement service" placement

openstack endpoint create --region RegionOne compute public http://172.26.15.175:8774/v2.1/%\(tenant\_id\)s

openstack endpoint create --region RegionOne compute internal http://172.26.15.175:8774/v2.1/%\(tenant\_id\)s

openstack endpoint create --region RegionOne compute admin http://172.26.15.175:8774/v2.1/%\(tenant\_id\)s

openstack endpoint create --region RegionOne placement public http://172.26.15.175:8778

openstack endpoint create --region RegionOne placement internal http://172.26.15.175:8778

openstack endpoint create --region RegionOne placement admin http://172.26.15.175:8778

---------------------------------------------------

root@dlp ~(keystone)# mysql

MariaDB [(none)]> create database nova; grant all privileges on nova.\* to nova@'localhost' identified by 'password'; grant all privileges on nova.\* to nova@'%' identified by 'password';

MariaDB [(none)]> create database nova\_api; grant all privileges on nova\_api.\* to nova@'localhost' identified by 'password'; grant all privileges on nova\_api.\* to nova@'%' identified by 'password';

MariaDB [(none)]> create database placement; grant all privileges on placement.\* to placement@'localhost' identified by 'password'; grant all privileges on placement.\* to placement@'%' identified by 'password';

MariaDB [(none)]> create database nova\_cell0; grant all privileges on nova\_cell0.\* to nova@'localhost' identified by 'password'; grant all privileges on nova\_cell0.\* to nova@'%' identified by 'password'; flush privileges;

MariaDB [(none)]> exit

Bye

---------------------------------------------------

2. Install the package and modify the configuration files

apt -y install nova-api nova-conductor nova-scheduler nova-novncproxy placement-api python3-novaclient

mv /etc/nova/nova.conf /etc/nova/nova.conf.bak

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/nova/nova.conf

# create new

[DEFAULT]

# define own IP address

my\_ip = 172.26.15.175

state\_path = /var/lib/nova

enabled\_apis = osapi\_compute,metadata

log\_dir = /var/log/nova

# RabbitMQ connection info

transport\_url = rabbit://openstack:password@172.26.15.175

[api]

auth\_strategy = keystone

# Glance connection info

[glance]

api\_servers = http://172.26.15.175:9292

[oslo\_concurrency]

lock\_path = $state\_path/tmp

# MariaDB connection info

[api\_database]

connection = mysql+pymysql://nova:password@172.26.15.175/nova\_api

[database]

connection = mysql+pymysql://nova:password@172.26.15.175/nova

# Keystone auth info

[keystone\_authtoken]

www\_authenticate\_uri = http://172.26.15.175:5000

auth\_url = http://172.26.15.175:5000

memcached\_servers = 172.26.15.175:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = nova

password = servicepassword

[placement]

auth\_url = http://172.26.15.175:5000

os\_region\_name = RegionOne

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = placement

password = servicepassword

[wsgi]

api\_paste\_config = /etc/nova/api-paste.ini

---------------------------------------------------

chmod 640 /etc/nova/nova.conf

chgrp nova /etc/nova/nova.conf

mv /etc/placement/placement.conf /etc/placement/placement.conf.bak

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/placement/placement.conf

# create new

[DEFAULT]

debug = false

[api]

auth\_strategy = keystone

[keystone\_authtoken]

www\_authenticate\_uri = http://172.26.15.175:5000

auth\_url = http://172.26.15.175:5000

memcached\_servers = 172.26.15.175:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = placement

password = servicepassword

[placement\_database]

connection = mysql+pymysql://placement:password@172.26.15.175/placement

# enable VNC

[vnc]

enabled = True

server\_listen = 0.0.0.0

server\_proxyclient\_address = 172.26.15.175

novncproxy\_base\_url = http://172.26.15.175:6080/vnc\_auto.html

---------------------------------------------------

chmod 640 /etc/placement/placement.conf

chgrp placement /etc/placement/placement.conf

su -s /bin/bash placement -c "placement-manage db sync"

su -s /bin/bash nova -c "nova-manage api\_db sync"

su -s /bin/bash nova -c "nova-manage cell\_v2 map\_cell0"

su -s /bin/bash nova -c "nova-manage db sync"

su -s /bin/bash nova -c "nova-manage cell\_v2 create\_cell --name cell1"

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/default/nova-consoleproxy

# line 6 : change

NOVA\_CONSOLE\_PROXY\_TYPE=novnc

---------------------------------------------------

systemctl restart apache2 nova-api nova-conductor nova-scheduler

systemctl enable apache2 nova-api nova-conductor nova-scheduler

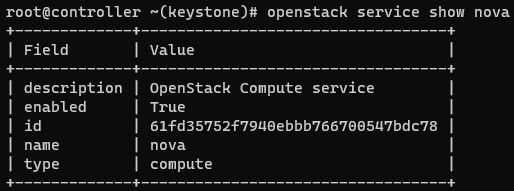
openstack compute service list

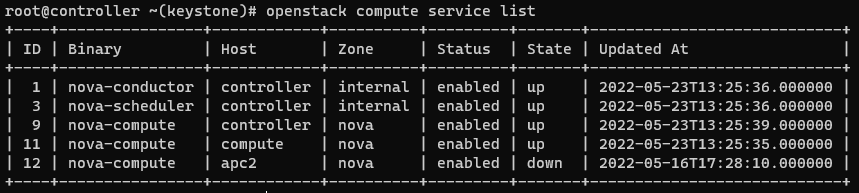
3. Also enable the controller node to act as a computing node at the same time, which means the instances in Openstack can be also running on this controller node. So you need to finish all the steps in the next chapter on this controller too. And after finishing it, run the following commands to detect all the nova nodes.

su -s /bin/bash nova -c "nova-manage cell\_v2 discover\_hosts"

openstack compute service list

4. After all the installation, you could check the nova service in the controller node.





### 3.2.4 Nova service (Computing Node)

1. Install qemu-kvm related components on the computing node and modify the grub file.

apt -y install qemu-kvm libvirt-daemon-system libvirt-daemon virtinst bridge-utils libosinfo-bin

apt -y install nova-compute nova-compute-kvm qemu-system-data

---------------------------------------------------

root@node01:~# vi /etc/default/grub

# line 10 : add

GRUB\_DEFAULT=0

GRUB\_TIMEOUT=5

GRUB\_DISTRIBUTOR=`lsb\_release -i -s 2> /dev/null || echo Debian`

GRUB\_CMDLINE\_LINUX\_DEFAULT="quiet"

GRUB\_CMDLINE\_LINUX="systemd.unified\_cgroup\_hierarchy=false systemd.legacy\_systemd\_cgroup\_controller=false"

---------------------------------------------------

update-grub

reboot

2. Install nova related packages, modify the configuration files and enable/start the nova services.

mv /etc/nova/nova.conf /etc/nova/nova.conf.bak

---------------------------------------------------

root@node01:~# vi /etc/nova/nova.conf

# create new

[DEFAULT]

# define own IP address

my\_ip = 172.26.15.223

state\_path = /var/lib/nova

enabled\_apis = osapi\_compute,metadata

log\_dir = /var/log/nova

# RabbitMQ connection info

transport\_url = rabbit://openstack:password@172.26.15.175

[api]

auth\_strategy = keystone

# enable VNC

[vnc]

enabled = True

server\_listen = 0.0.0.0

server\_proxyclient\_address = $my\_ip

novncproxy\_base\_url = http://172.26.15.175:6080/vnc\_auto.html

# Glance connection info

[glance]

api\_servers = http://172.26.15.175:9292

[oslo\_concurrency]

lock\_path = $state\_path/tmp

# Keystone auth info

[keystone\_authtoken]

www\_authenticate\_uri = http://172.26.15.175:5000

auth\_url = http://172.26.15.175:5000

memcached\_servers = 172.26.15.175:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = nova

password = servicepassword

[placement]

auth\_url = http://172.26.15.175:5000

os\_region\_name = RegionOne

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = placement

password = servicepassword

[wsgi]

api\_paste\_config = /etc/nova/api-paste.ini

---------------------------------------------------

chmod 640 /etc/nova/nova.conf

chgrp nova /etc/nova/nova.conf

systemctl restart nova-compute

systemctl enable nova-compute

### 3.2.5 Neutron service (Controller Node)

1. Configure Openstack and mysql user/permission for nova service

openstack user create --domain default --project service --password servicepassword neutron

openstack role add --project service --user neutron admin

openstack service create --name neutron --description "OpenStack Networking service" network

openstack endpoint create --region RegionOne network public http://172.26.15.175:9696

openstack endpoint create --region RegionOne network internal http://172.26.15.175:9696

openstack endpoint create --region RegionOne network admin http://172.26.15.175:9696

---------------------------------------------------

root@dlp ~(keystone)# mysql

MariaDB [(none)]> create database neutron\_ml2; grant all privileges on neutron\_ml2.\* to neutron@'localhost' identified by 'password'; grant all privileges on neutron\_ml2.\* to neutron@'%' identified by 'password'; flush privileges;

MariaDB [(none)]> exit

Bye

---------------------------------------------------

2. Install neutron related packages, modify the configuration files and enable/start the nova services. (Here, I chose to configure “OpenvSwitch + Local/Vlan/Vxlan”)

apt -y install neutron-server neutron-metadata-agent neutron-plugin-ml2 python3-neutronclient

mv /etc/neutron/neutron.conf /etc/neutron/neutron.conf.bak

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/neutron/neutron.conf

# create new

[DEFAULT]

core\_plugin = ml2

service\_plugins = router

auth\_strategy = keystone

state\_path = /var/lib/neutron

dhcp\_agent\_notification = True

allow\_overlapping\_ips = True

notify\_nova\_on\_port\_status\_changes = True

notify\_nova\_on\_port\_data\_changes = True

# RabbitMQ connection info

transport\_url = rabbit://openstack:password@172.26.15.175

[agent]

root\_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf

# Keystone auth info

[keystone\_authtoken]

www\_authenticate\_uri = http://172.26.15.175:5000

auth\_url = http://172.26.15.175:5000

memcached\_servers = 172.26.15.175:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = neutron

password = servicepassword

# MariaDB connection info

[database]

connection = mysql+pymysql://neutron:password@172.26.15.175/neutron\_ml2

# Nova connection info

[nova]

auth\_url = http://172.26.15.175:5000

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

region\_name = RegionOne

project\_name = service

username = nova

password = servicepassword

[oslo\_concurrency]

lock\_path = $state\_path/tmp

---------------------------------------------------

chmod 640 /etc/neutron/neutron.conf

chgrp neutron /etc/neutron/neutron.conf

cp /etc/neutron/metadata\_agent.ini /etc/neutron/metadata\_agent.ini.bak

cp /etc/neutron/plugins/ml2/ml2\_conf.ini /etc/neutron/plugins/ml2/ml2\_conf.ini.bak

---------------------------------------------------

root@controller neutron(keystone)# grep -v "^#" /etc/neutron/metadata\_agent.ini | grep -v "^$"

[DEFAULT]

nova\_metadata\_host = 172.26.15.175

metadata\_proxy\_shared\_secret = metadata\_secret

[agent]

[cache]

memcache\_servers = 172.26.15.175:11211

root@controller neutron(keystone)# grep -v "^#" /etc/neutron/plugins/ml2/ml2\_conf.ini | grep -v "^$"

[DEFAULT]

[ml2]

type\_drivers = local,flat,vlan

tenant\_network\_types = local,flat,vlan

mechanism\_drivers = openvswitch,l2population

extension\_drivers = port\_security,qos

[ml2\_type\_flat]

flat\_networks = external

[ml2\_type\_geneve]

[ml2\_type\_gre]

[ml2\_type\_vlan]

[ml2\_type\_vxlan]

vni\_ranges = 1:1000

[ovs\_driver]

[securitygroup]

enable\_security\_group = True

enable\_ipset = True

[sriov\_driver]

root@controller neutron(keystone)# grep -v "^#" /etc/neutron/plugins/ml2/openvswitch\_agent.ini | grep -v "^$"

[DEFAULT]

[agent]

tunnel\_types = vxlan

[network\_log]

[ovs]

integration\_bridge = br-int

tunnel\_bridge = br-tun

bridge\_mappings = external:br-ex

[securitygroup]

firewall\_driver = openvswitch

enable\_security\_group = True

enable\_ipset = True

[xenapi]

root@controller neutron(keystone)# grep -v "^#" /etc/neutron/dhcp\_agent.ini | grep -v "^$"

[DEFAULT]

interface\_driver = openvswitch

enable\_isolated\_metadata = True

[agent]

[ovs]

root@controller neutron(keystone)# grep -v "^#" /etc/neutron/l3\_agent.ini | grep -v "^$"

[DEFAULT]

ovs\_use\_veth = False

interface\_driver = openvswitch

[agent]

[network\_log]

[ovs]

---------------------------------------------------

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/nova/nova.conf

# add follows into [DEFAULT] section

use\_neutron = True

# add follows to the end : Neutron auth info

# the value of [metadata\_proxy\_shared\_secret] is the same with the one in [metadata\_agent.ini]

[neutron]

auth\_url = http://172.26.15.175:5000

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

region\_name = RegionOne

project\_name = service

username = neutron

password = servicepassword

service\_metadata\_proxy = True

metadata\_proxy\_shared\_secret = metadata\_secret

---------------------------------------------------

ln -s /etc/neutron/plugins/ml2/ml2\_conf.ini /etc/neutron/plugin.ini

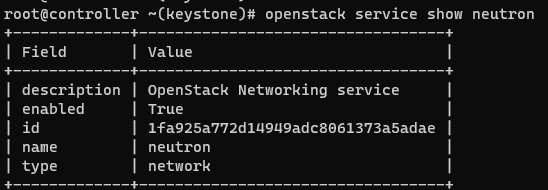
su -s /bin/bash neutron -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugin.ini upgrade head"

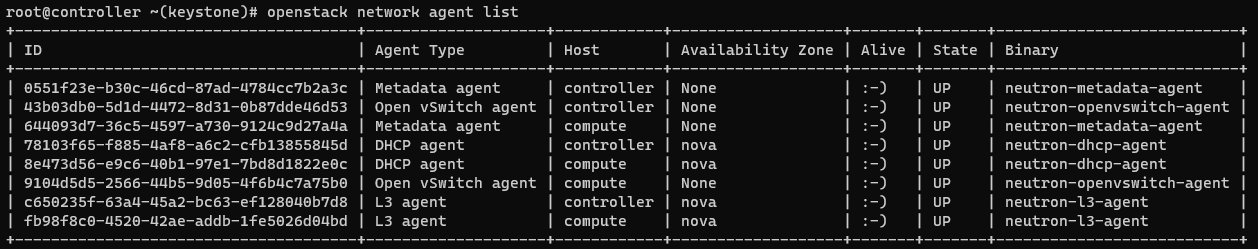
systemctl restart neutron-api neutron-rpc-server neutron-metadata-agent nova-api

systemctl enable neutron-api neutron-rpc-server neutron-metadata-agent nova-api

3. As the controller node also works as the computing node at the same time, we also need to configure the network service for the controller node, thus, we need to finish the steps in the next chapter for controller node too.

4. After installation, you could check neutron related services on the controller node of Openstack.





### 3.2.6 Neutron service (Computing Node)

1. Install neutron related packages, modify the configuration files and enable/start the nova services. (Here, I chose to configure “OpenvSwitch + Local/Vlan/Vxlan”)

apt -y install neutron-common neutron-plugin-ml2 neutron-openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent python3-neutronclient

mv /etc/neutron/neutron.conf /etc/neutron/neutron.conf.bak

---------------------------------------------------

vi /etc/neutron/neutron.conf

# create new

[DEFAULT]

core\_plugin = ml2

service\_plugins = router

auth\_strategy = keystone

state\_path = /var/lib/neutron

allow\_overlapping\_ips = True

# RabbitMQ connection info

transport\_url = rabbit://openstack:password@172.26.15.175

[agent]

root\_helper = sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf

# Keystone auth info

[keystone\_authtoken]

www\_authenticate\_uri = http://172.26.15.175:5000

auth\_url = http://172.26.15.175:5000

memcached\_servers = 172.26.15.175:11211

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

project\_name = service

username = neutron

password = servicepassword

[oslo\_concurrency]

lock\_path = $state\_path/lock

---------------------------------------------------

chmod 640 /etc/neutron/neutron.conf

chgrp neutron /etc/neutron/neutron.conf

cp /etc/neutron/l3\_agent.ini /etc/neutron/l3\_agent.ini.bak

cp /etc/neutron/dhcp\_agent.ini /etc/neutron/dhcp\_agent.ini.bak

cp /etc/neutron/metadata\_agent.ini /etc/neutron/metadata\_agent.ini.bak

cp /etc/neutron/plugins/ml2/ml2\_conf.ini /etc/neutron/plugins/ml2/ml2\_conf.ini.bak

cp /etc/neutron/plugins/ml2/openvswitch\_agent.ini /etc/neutron/plugins/ml2/openvswitch\_agent.ini.bak

---------------------------------------------------

root@compute:~# grep -v "^#" /etc/neutron/metadata\_agent.ini | grep -v "^$"

[DEFAULT]

nova\_metadata\_host = 172.26.15.175

metadata\_proxy\_shared\_secret = metadata\_secret

[agent]

[cache]

memcache\_servers = 172.26.15.175:11211

root@compute:~# grep -v "^#" /etc/neutron/plugins/ml2/ml2\_conf.ini | grep -v "^$"

[DEFAULT]

[ml2]

type\_drivers = local,flat,vlan

tenant\_network\_types = local,flat,vlan

mechanism\_drivers = openvswitch,l2population

extension\_drivers = port\_security,qos

[ml2\_type\_flat]

flat\_networks = external

[ml2\_type\_geneve]

[ml2\_type\_gre]

[ml2\_type\_vlan]

[ml2\_type\_vxlan]

vni\_ranges = 1:1000

[ovs\_driver]

[securitygroup]

enable\_security\_group = True

enable\_ipset = True

[sriov\_driver]

181 ovs-vsctl add-br br-ex

183 ovs-vsctl add-port br-ex eno2

root@compute:~# grep -v "^#" /etc/neutron/plugins/ml2/openvswitch\_agent.ini | grep -v "^$"

[DEFAULT]

[agent]

tunnel\_types = vxlan

[network\_log]

[ovs]

integration\_bridge = br-int

tunnel\_bridge = br-tun

local\_ip = 172.26.15.223

bridge\_mappings = external:br-ex

[securitygroup]

firewall\_driver = openvswitch

enable\_security\_group = True

enable\_ipset = True

[xenapi]

root@compute:~# grep -v "^#" /etc/neutron/l3\_agent.ini | grep -v "^$"

[DEFAULT]

ovs\_use\_veth = True

interface\_driver = openvswitch

[agent]

[network\_log]

[ovs]

root@compute:~# grep -v "^#" /etc/neutron/dhcp\_agent.ini | grep -v "^$"

[DEFAULT]

interface\_driver = openvswitch

dhcp\_driver = neutron.agent.linux.dhcp.Dnsmasq

enable\_isolated\_metadata = True

[agent]

[ovs]

---------------------------------------------------

---------------------------------------------------

root@node01:~# vi /etc/nova/nova.conf

# add follows into [DEFAULT] section

use\_neutron = True

vif\_plugging\_is\_fatal = True

vif\_plugging\_timeout = 300

# add follows to the end: Neutron auth info

# the value of [metadata\_proxy\_shared\_secret] is the same with the one in [metadata\_agent.ini]

[neutron]

auth\_url = http://172.26.15.175:5000

auth\_type = password

project\_domain\_name = default

user\_domain\_name = default

region\_name = RegionOne

project\_name = service

username = neutron

password = servicepassword

service\_metadata\_proxy = True

metadata\_proxy\_shared\_secret = metadata\_secret

---------------------------------------------------

ln -s /etc/neutron/plugins/ml2/ml2\_conf.ini /etc/neutron/plugin.ini

systemctl restart nova-compute neutron-openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent

systemctl enable nova-compute neutron-openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent

### 3.2.7 Security group and flavour configuration (Controller Node)

openstack security group create secgroup01

openstack security group list

ssh-keygen -q -N ""

openstack keypair create --public-key ~/.ssh/id\_rsa.pub mykey

openstack keypair list

openstack project create --domain default --description "Hiroshima Project" hiroshima

openstack user create --domain default --project hiroshima --password userpassword serverworld

openstack role create CloudUser

openstack role add --project hiroshima --user serverworld CloudUser

openstack flavor create --id 0 --vcpus 1 --ram 2048 --disk 10 m1.small

### 3.2.8 Horizon service (Controller Node)

1. Install herozon related packages, modify the configuration files and enable/start the services.

apt -y install openstack-dashboard

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/openstack-dashboard/local\_settings.py

# line 40 : specify allowed compute hosts to connect to horizon

ALLOWED\_HOSTS = ['\*']

# line 107 : add

SESSION\_ENGINE = "django.contrib.sessions.backends.cache"

# line 120 : set Openstack Host

# line 121 : comment out and add a line to specify URL of Keystone Host

OPENSTACK\_HOST = "172.26.15.175"

OPENSTACK\_KEYSTONE\_URL = "http://172.26.15.175:5000/v3"

---------------------------------------------------

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/openstack-dashboard/local\_settings.d/\_0006\_debian\_cache.py

# change to your Memcache server

CACHES = {

'default' : {

'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',

'LOCATION': '172.26.15.175:11211',

}

}

---------------------------------------------------

---------------------------------------------------

root@dlp ~(keystone)# vi /etc/apache2/conf-available/openstack-dashboard.conf

# create new

WSGIScriptAlias / /usr/share/openstack-dashboard/wsgi.py process-group=horizon

WSGIDaemonProcess horizon user=horizon group=horizon processes=3 threads=10 display-name=%{GROUP}

WSGIProcessGroup horizon

WSGIApplicationGroup %{GLOBAL}

Alias /static /var/lib/openstack-dashboard/static/

Alias /horizon/static /var/lib/openstack-dashboard/static/

<Directory /usr/share/openstack-dashboard>

Require all granted

</Directory>

<Directory /var/lib/openstack-dashboard/static>

Require all granted

</Directory>

---------------------------------------------------

a2enconf openstack-dashboard; systemctl reload apache2; a2enconf openstack-dashboard

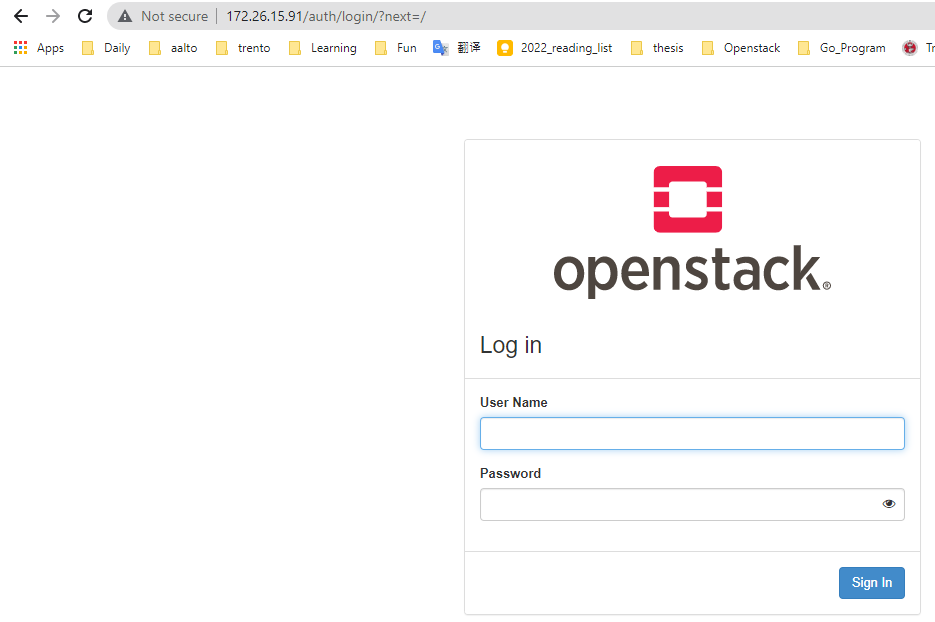
mv /etc/openstack-dashboard/policy /etc/openstack-dashboard/policy.bak

chown -R horizon /var/lib/openstack-dashboard/secret-key

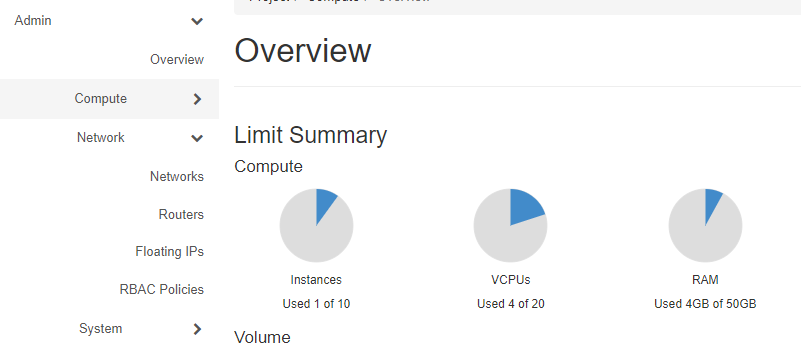
systemctl restart apache2

### 3.2.9 Start the instance through web UI

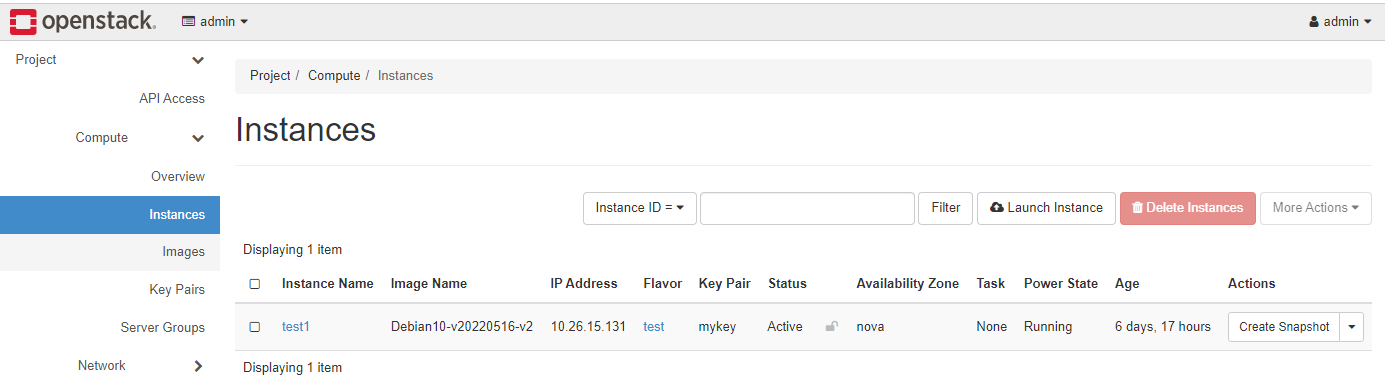
Now you can login the openstack with admin/adminpassword



Configure the Networks and Routers which will be used by the instances you create later.



Create instance



# 4. How to setup Framework 4 (Kubernetes) on APC 910

## 4.1 Introduction of Framework 4

The following figure shows the architecture of Framework4, if you are not familiar with both OpenStack and Kubernetes, pls google it, there are lots of open source materials you could access in the Internet.

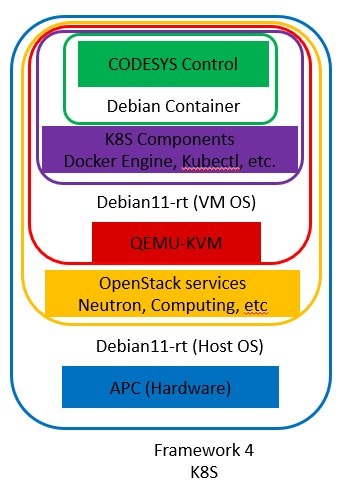


Figure 10: Architecture of framework 4

## 4.2 Steps to setup Framework 4

**Important**: I did not finish the deployment of framework 4, but succeeded in running CODESYS control process in a debian-based container, also finished the deployment of Kubernetes, the reason I did not deploy framework4 is caused by the limitation of CODESYS software. As you can see from the following picture, which shows how a CODESYS IDE running on Windows connects to the Linux-based machine to run the CODESYS Control process. The default method of the connection is SSH protocol and using the default 22 port. If I deploy Kubernetes on OpenStack, The containers managed by Kubernetes will not be accessed from outside through IP:22 easily. At least, I could not solve this issue in a pretty short time.

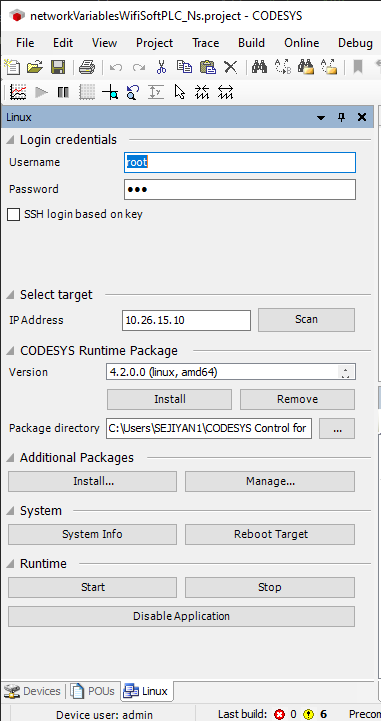


Figure 11: How a CODESYS IDE running on a Windows connects to a Linux-based machine to run CODESYS Control

### 4.2.1 How to deploy Kubernetes

Install the necessary packages, initiate a cluster, and configure the network

**# Install kubectl and kubectl-convert**

apt-get update

apt-get install -y apt-transport-https ca-certificates curl

curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubectl

source /usr/share/bash-completion/bash\_completion

echo 'source /usr/share/bash-completion/bash\_completion' >> ~/.bashrc

echo 'export KUBECONFIG=/etc/kubernetes/admin.conf' >> ~/.bashrc

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl-convert"

sudo install -o root -g root -m 0755 kubectl-convert /usr/local/bin/kubectl-convert

**# Install container runtime (docker-engine)**

apt-get update

apt-get install ca-certificates curl gnupg lsb-release -y

curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

apt-get update

apt-get install docker-ce docker-ce-cli containerd.io -y

systemctl restart docker.service; systemctl enable docker.service

**# Install kubeadm**

apt-get update

apt-get install -y apt-transport-https ca-certificates curl

curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm

sudo apt-mark hold kubelet kubeadm

**# Install a single control-plane Kubernetes cluster**

kubeadm init --pod-network-cidr=10.244.1.0/24

"By default, your cluster will not schedule Pods on the control-plane node for security reasons. If you want to be able to schedule Pods on the control-plane node, for example for a single-machine Kubernetes cluster for development, run:"

kubectl taint nodes --all node-role.kubernetes.io/master-

curl -O https://raw.githubusercontent.com/kvaps/bridget/master/bridget.yaml

kubectl create -f bridget.yaml

# https://github.com/kvaps/bridget

kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml

# https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/#internal-to-service

### 4.2.2 How to make a Docker image for CODESYS Control

1. In this step, you could just use Framework 1 to finish the deployment. Install docker engine first, or you could follow the following link:

<https://docs.docker.com/engine/install/debian/>

2. Configure the network for docker service, br0 is the bridge network which connects to the physical network interface on this machine. Then restart docker related services.

More explanation here: To make the containers have the IP address which can be accessed by outside instead of using “docker0” default network created by docker service.

# /etc/docker/daemon.json

{

"bridge": "br0",

"iptables": false

}

# systemctl restart docker

3. Build the docker image from the following Dockerfile.

FROM debian:latest

RUN apt update && apt install openssh-server -y

RUN echo "Port 22" >> /etc/ssh/sshd\_config

RUN echo "PermitRootLogin yes" >> /etc/ssh/sshd\_config

RUN echo "PubkeyAuthentication yes" >> /etc/ssh/sshd\_config

RUN echo "PasswordAuthentication yes" >> /etc/ssh/sshd\_config

RUN echo 'root:123' | chpasswd

RUN service ssh start

EXPOSE 22

CMD ["/usr/sbin/sshd","-D"]

# docker build -t debianssh:v1 .

# docker image ls  
REPOSITORY TAG IMAGE ID CREATED SIZE  
debianssh v1 32215b7c6284 17 seconds ago 181MB  
debian latest c4905f2a4f97 2 weeks ago 124MB

# docker run -it -d --name debianssh1 debianssh:v1

4. Start the container from the image you made above.

# 5. Connection between the frameworks and the IoT motion control target

## 5.1 Introduction of the connection between the frameworks and IoT motion control target

On the left side, a CODESYS IDE (Integrated Development Environment) is running and communicating with the CODESYS control process (soft PLC (PROGRAMMABLE LOGIC CONTROLLER)) which is running in the Debian 11 Host, Debian 11 VM or a Debian based container, with Ethernet, WIFI or 5G, the softPLC could control the RevPi on the right side as follows.

The CODESYS Control on the left side will be migrated to our edge computing platform and run there.

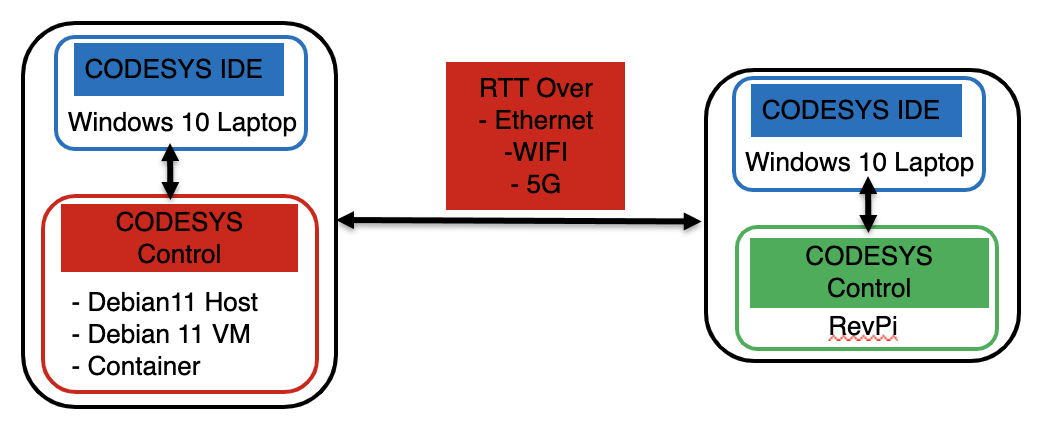


Figure 12: Rough architecture of IoT motion control system

## 5.2 CODESYS Program to test the connection between frameworks and the IoT motion control target

You could use the program in the following link to test the RTT between a few frameworks and the RevPi.

<https://abb-my.sharepoint.com/:f:/p/pang_zhibo_se/EiTSyI-NKZNEpO5h_6sk0TUBiFxvGGpQWhFx2XSzM60eNg?e=2HjfbU>

## 5.3 How to Create Windows 10 Image for OpenStack VM

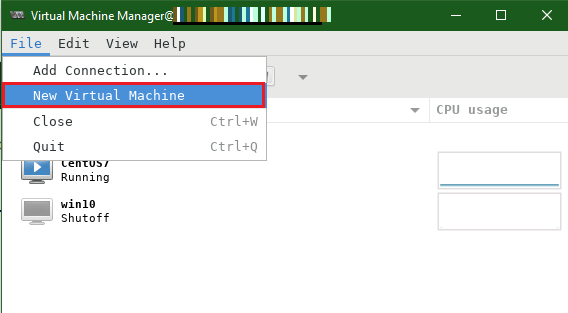
### 5.3.1 Introduction about the background

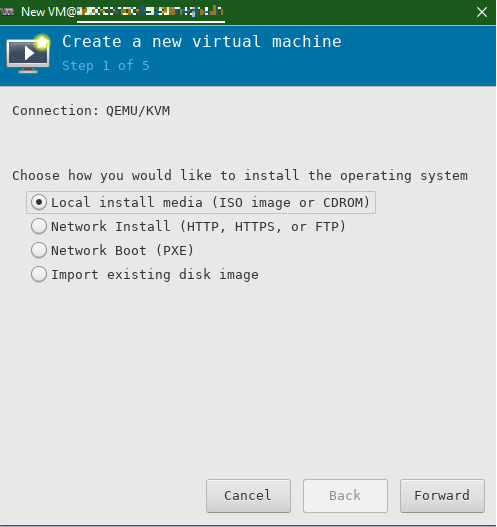
As CODESYS IDE needs to be running in Windows 10 environment, if you want to run the CODESYS IDE in OpenStack platform, then you need to make a Windows 10 image to start a Windows 10 VM in OpenStack.

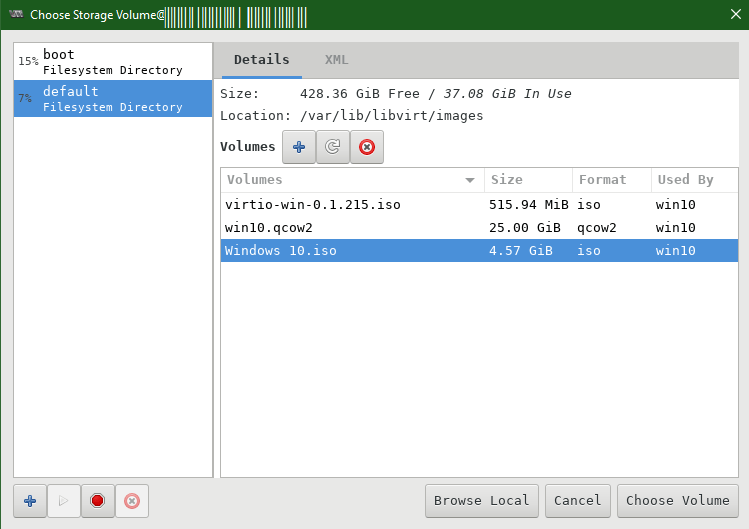
### 5.3.2 Steps to make Window 10 Image for OpenStack VM

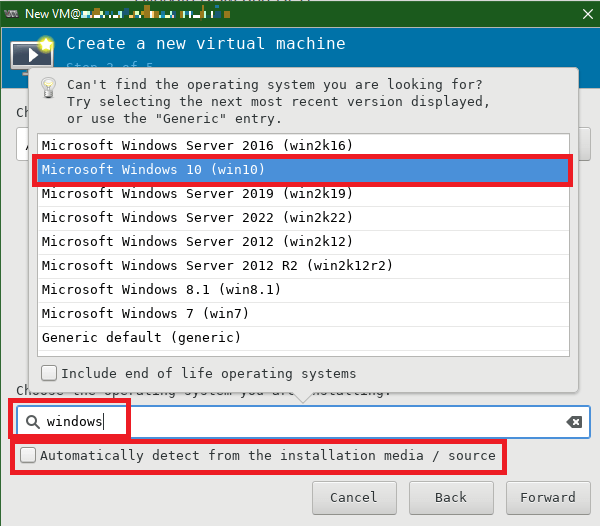
1. You can use the Framework 1, install “virt-manager” package and run the command “virt-manager”, navigate to File -> New Virtual Machine.

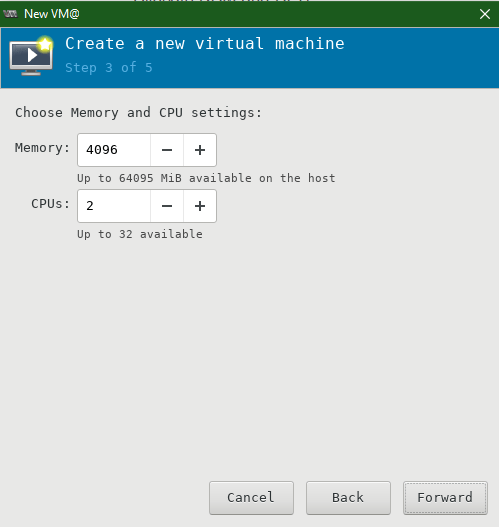
# virt-manager

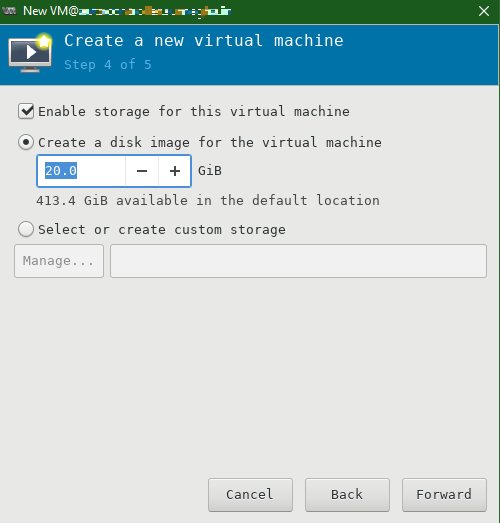


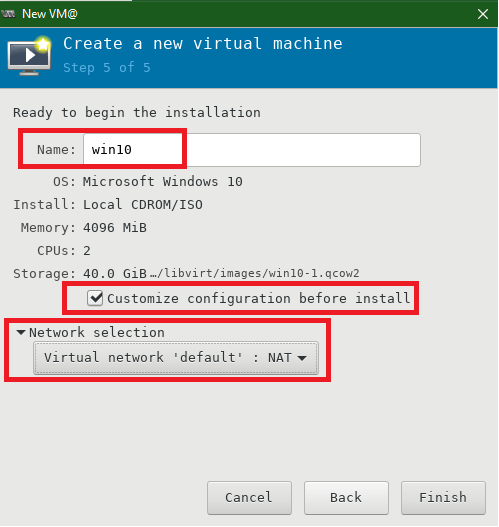


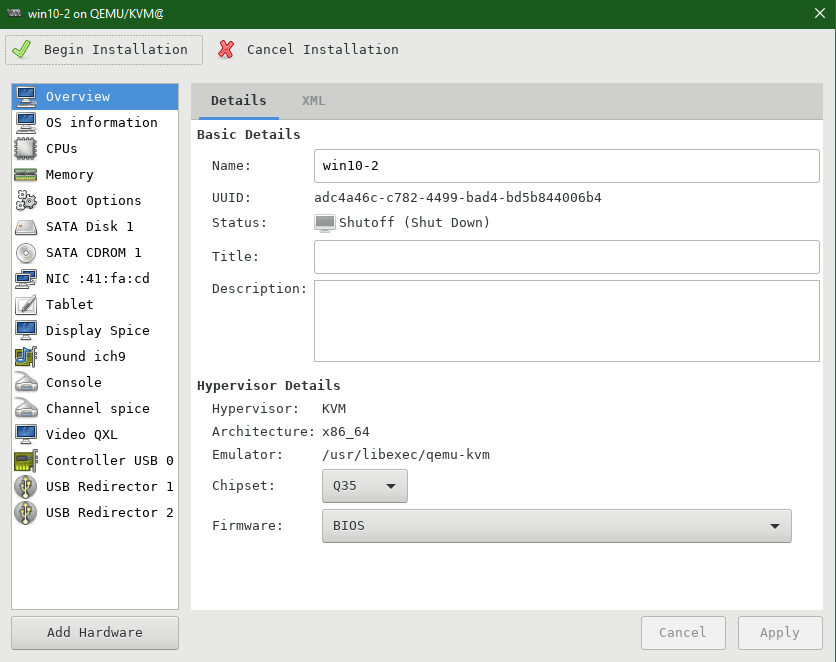


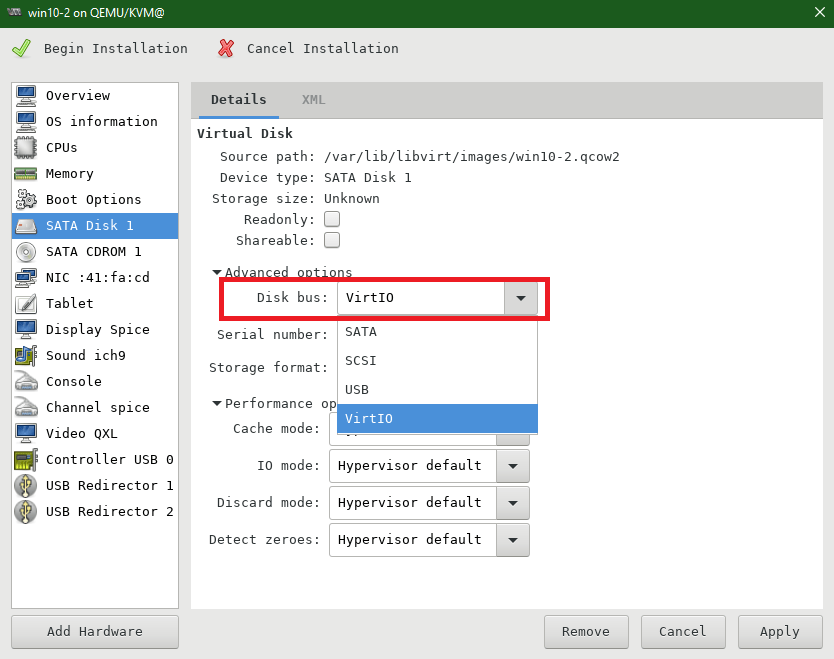


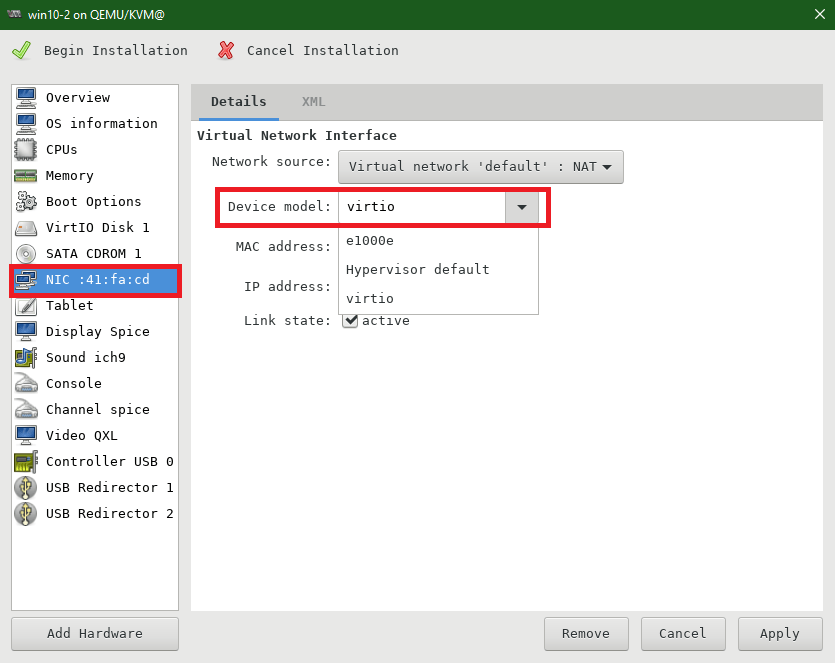


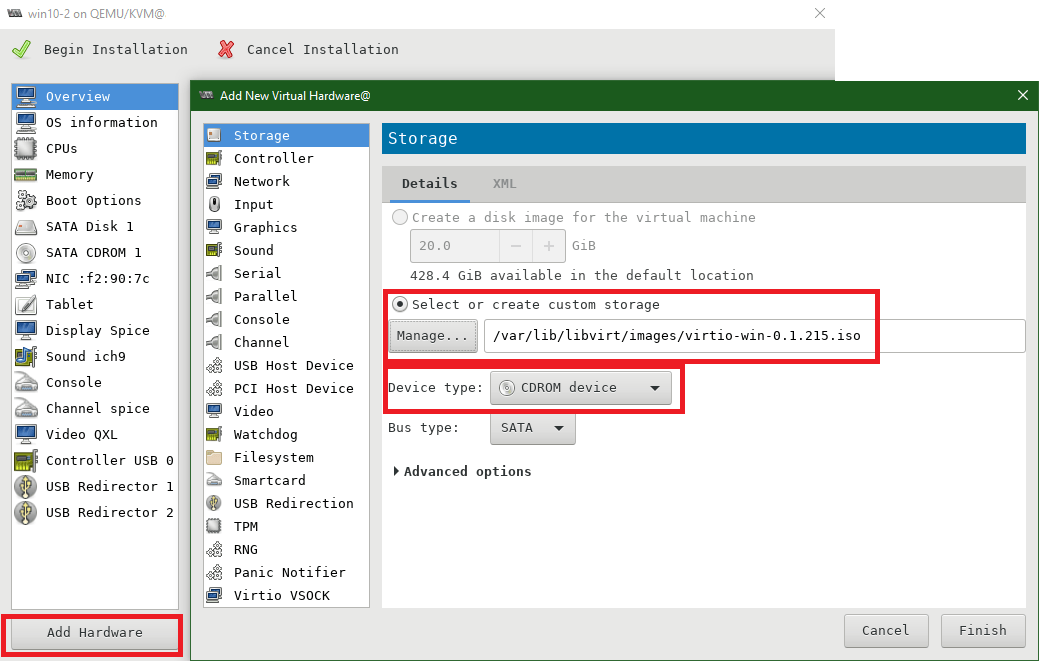


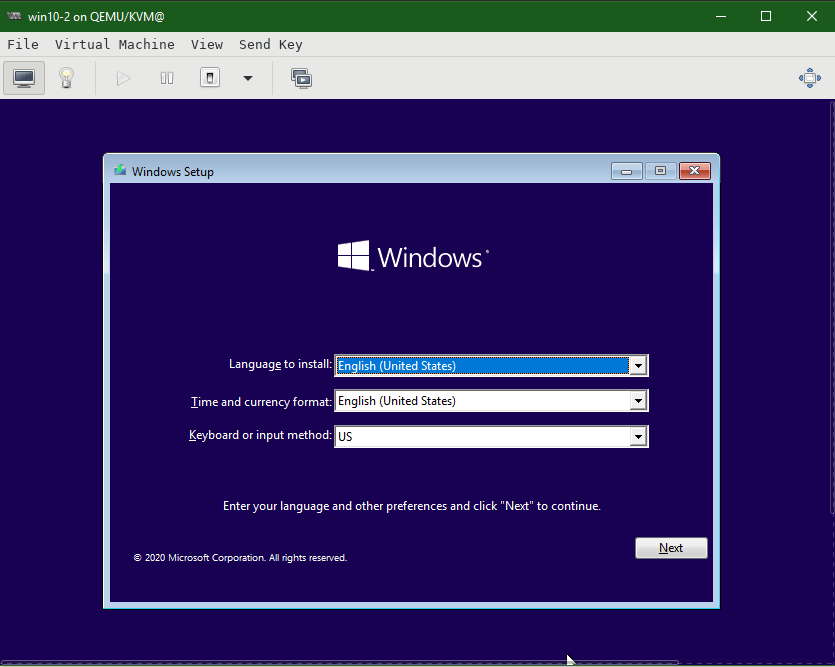


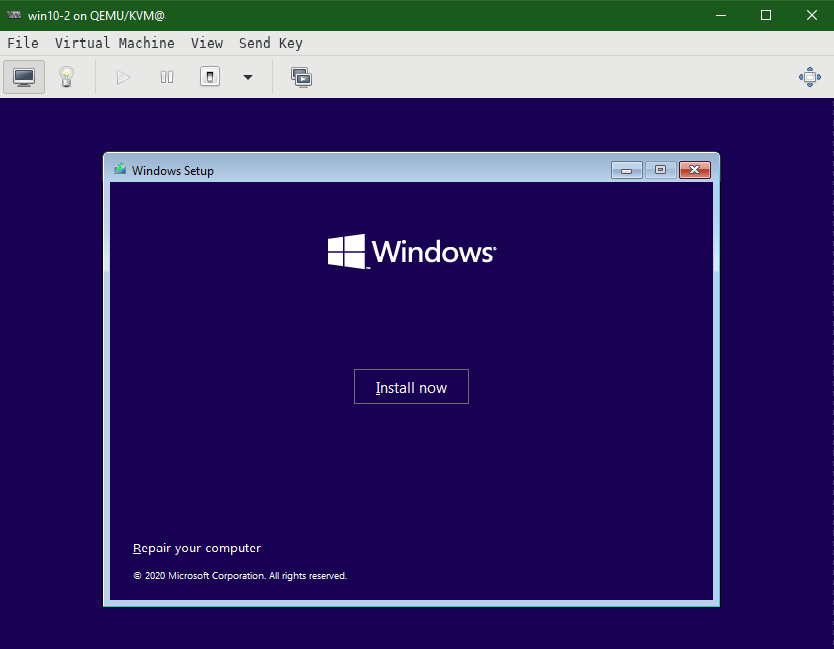


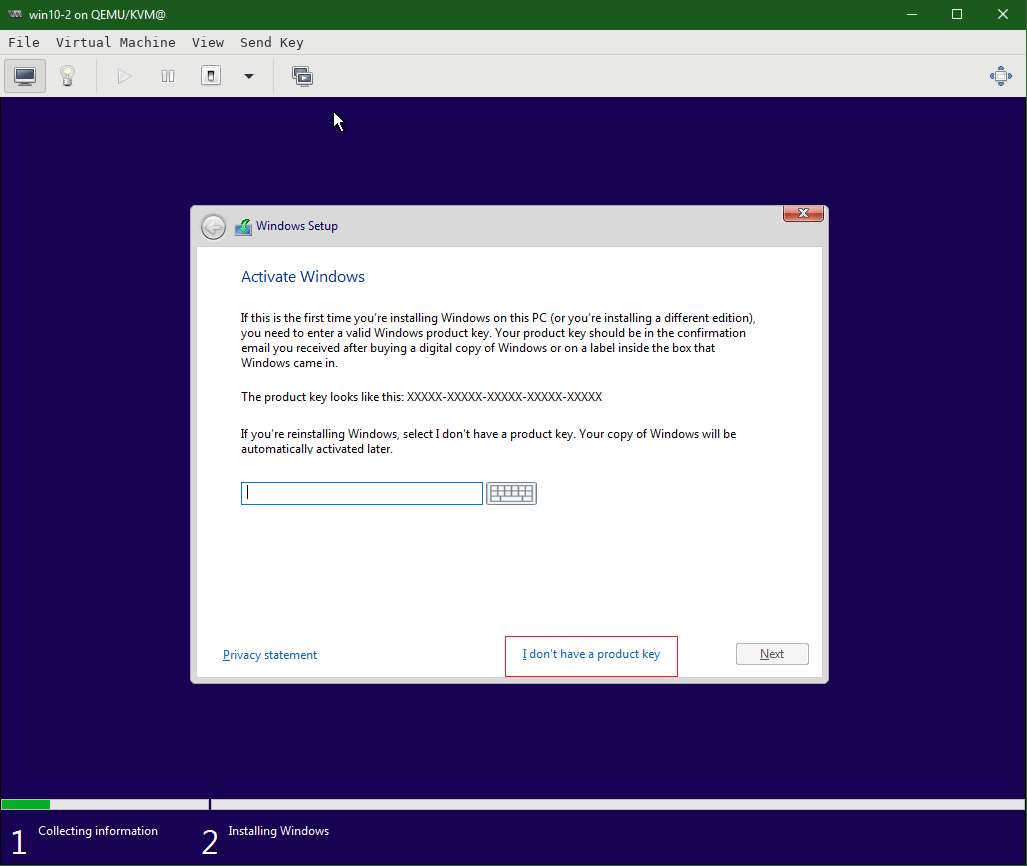


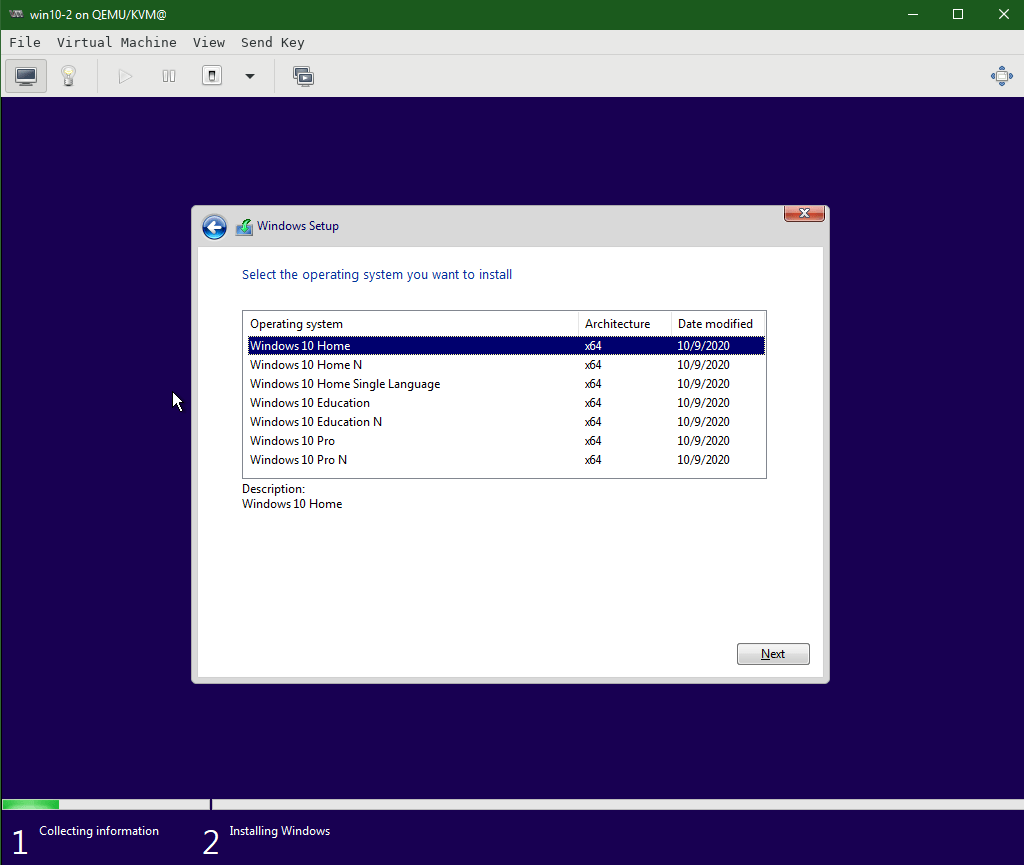


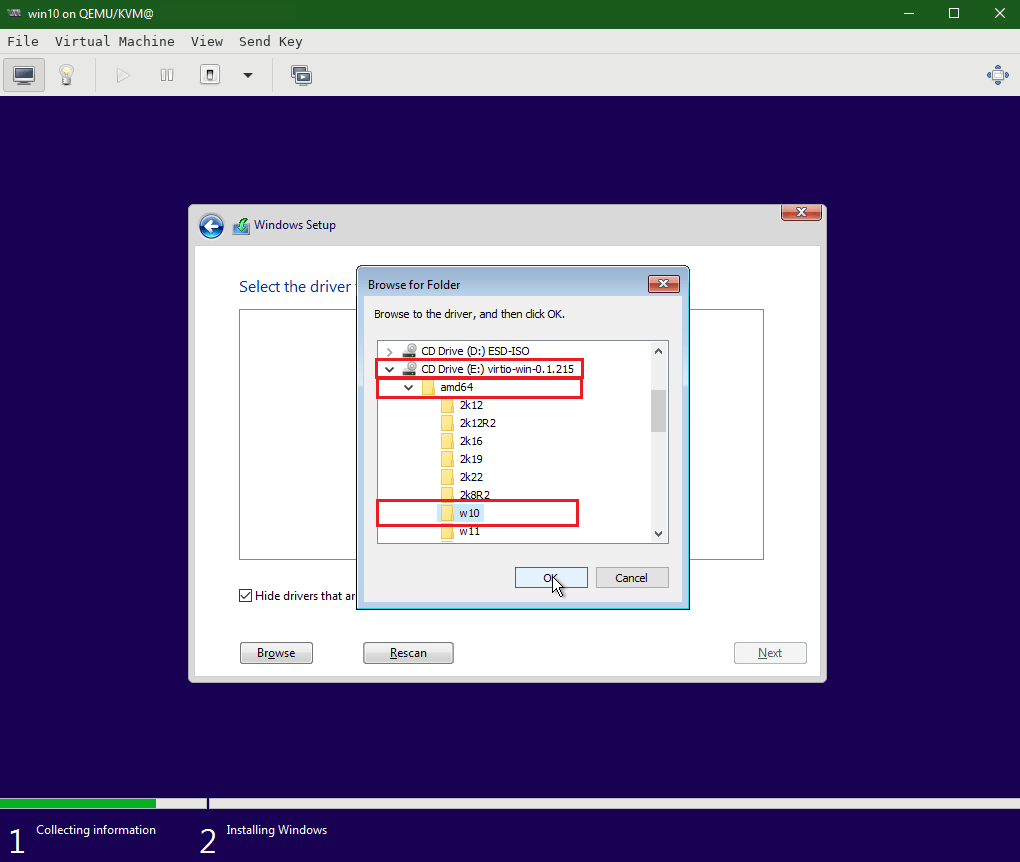
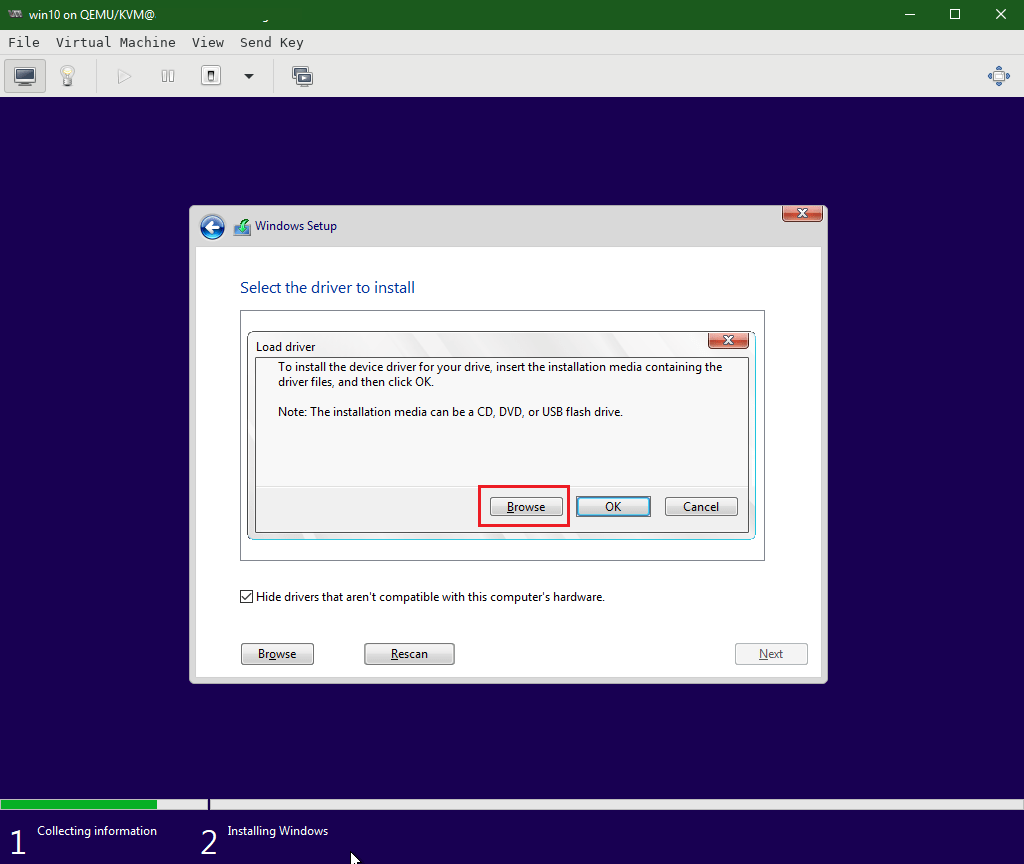
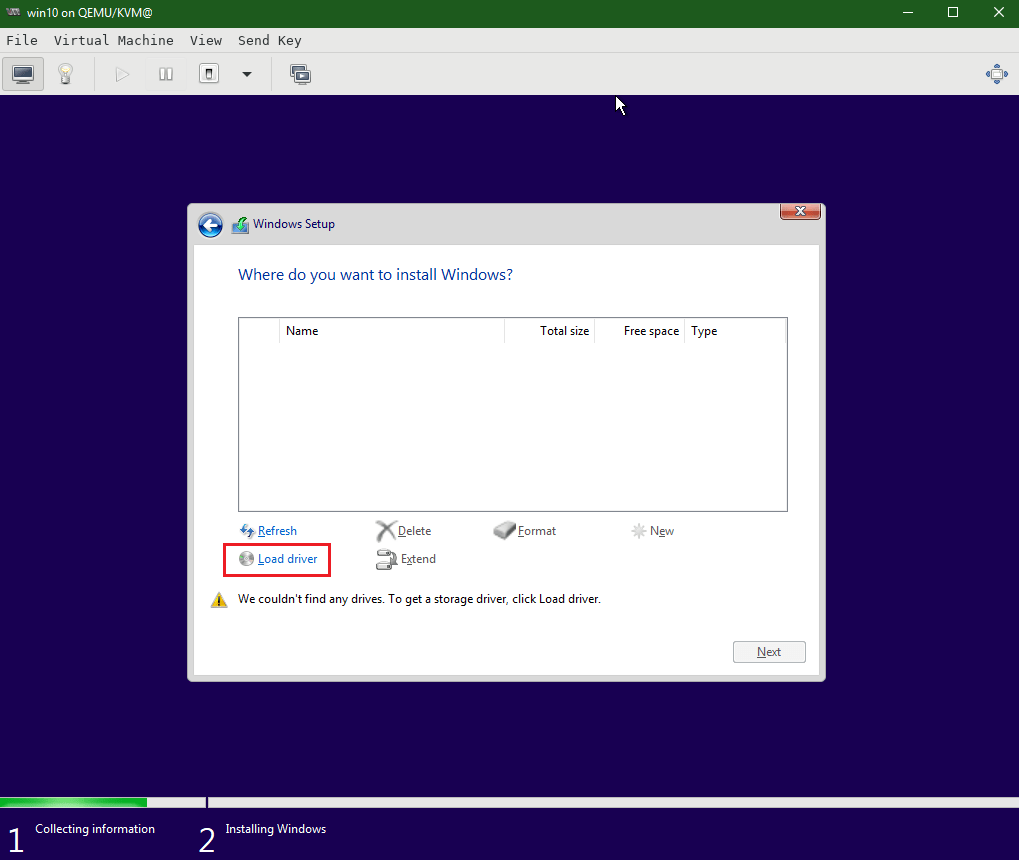


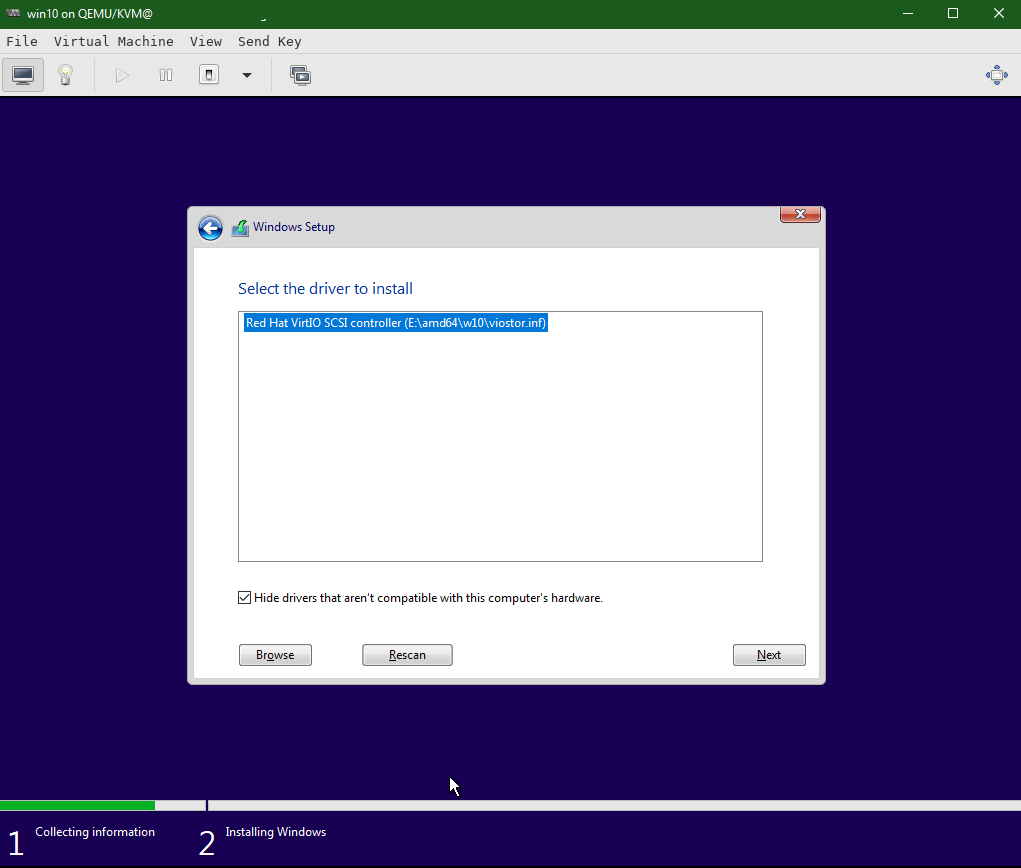


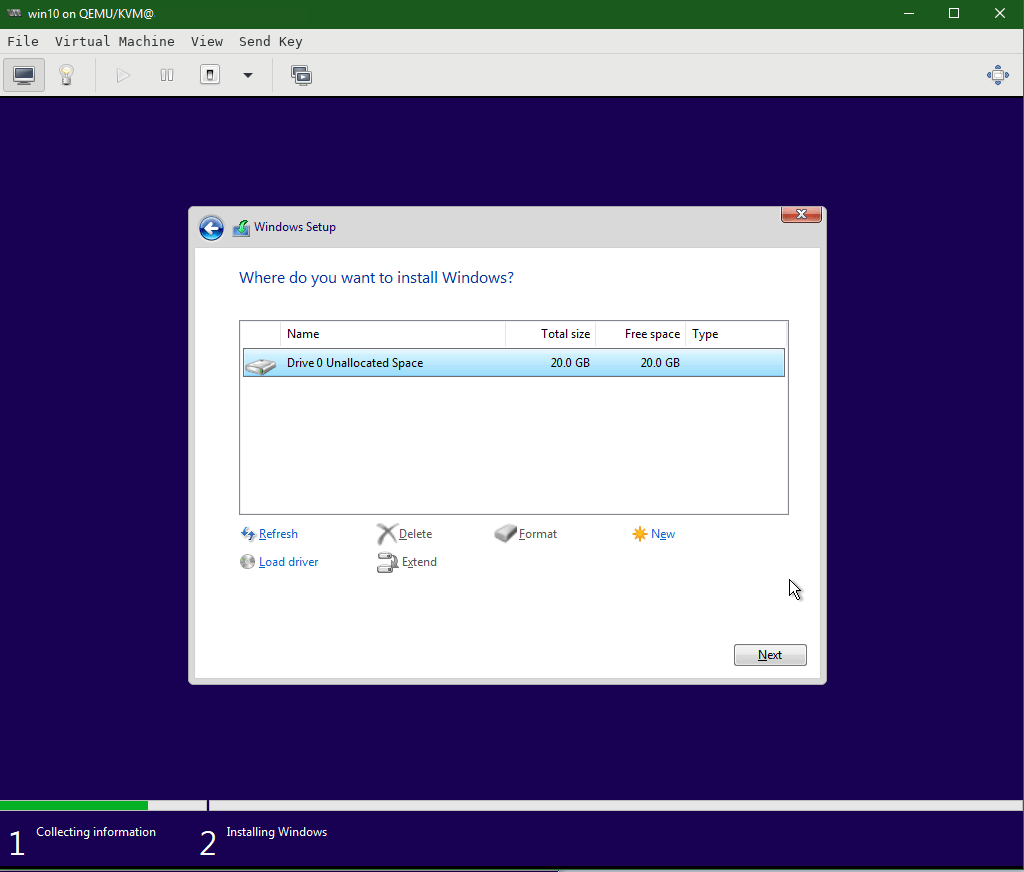


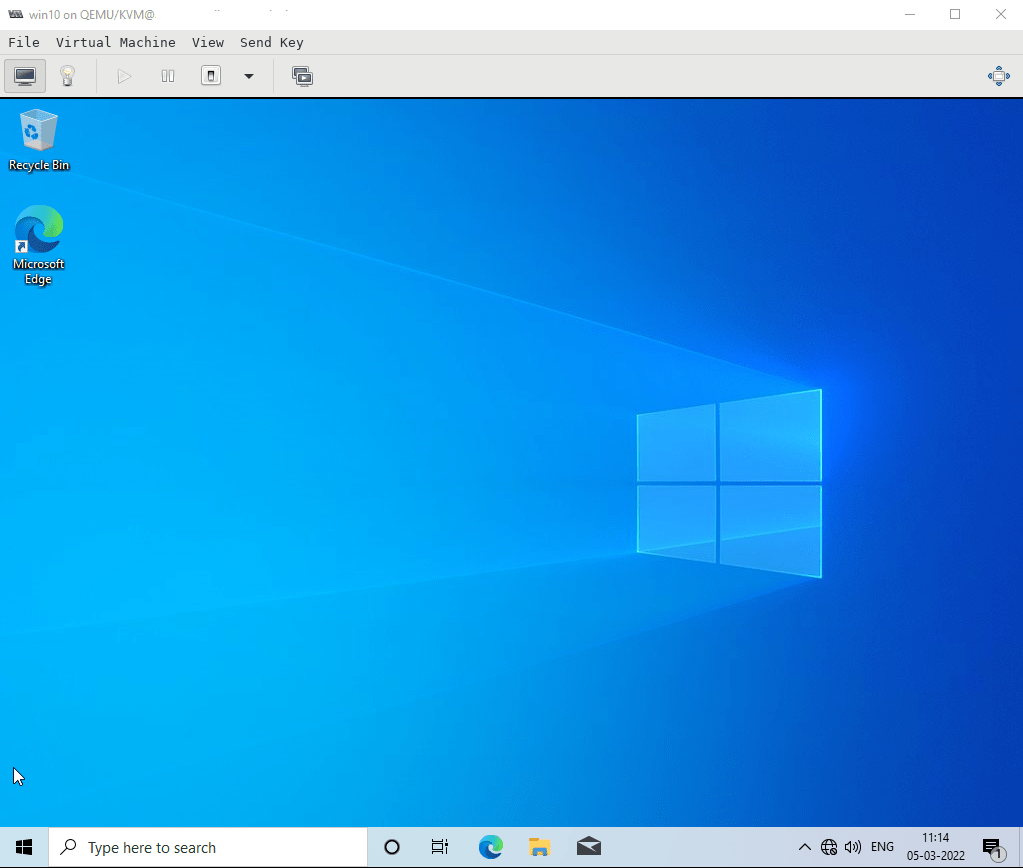


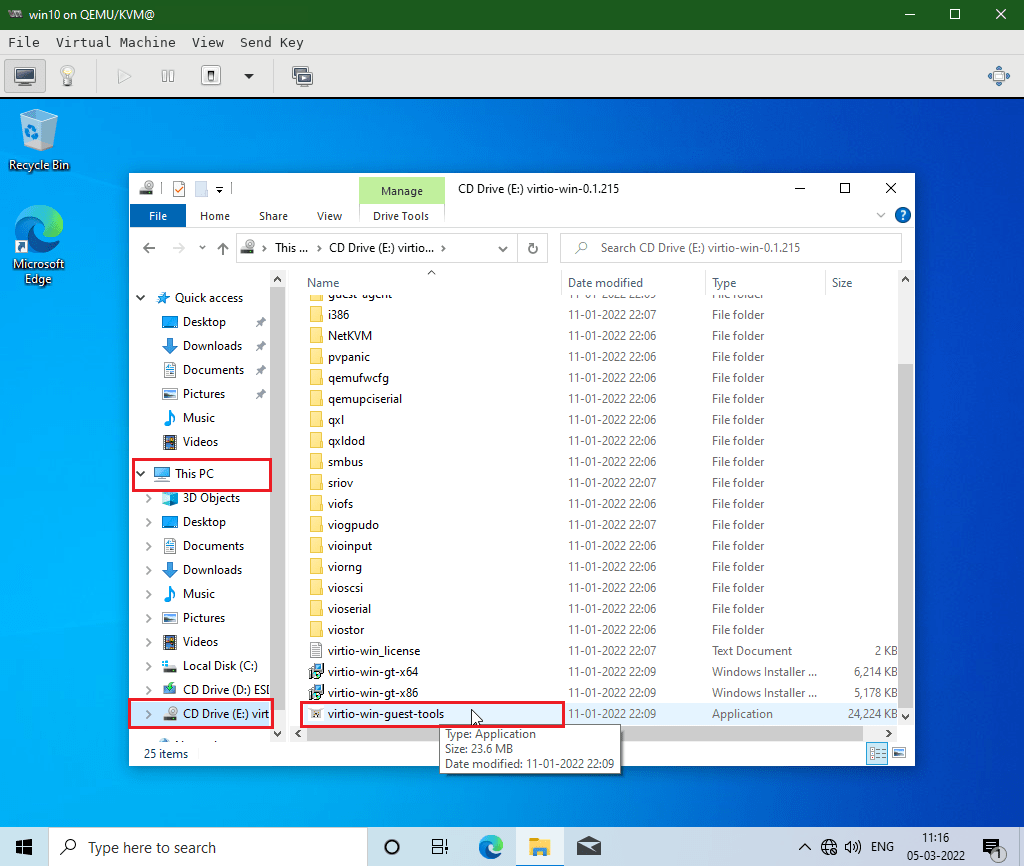


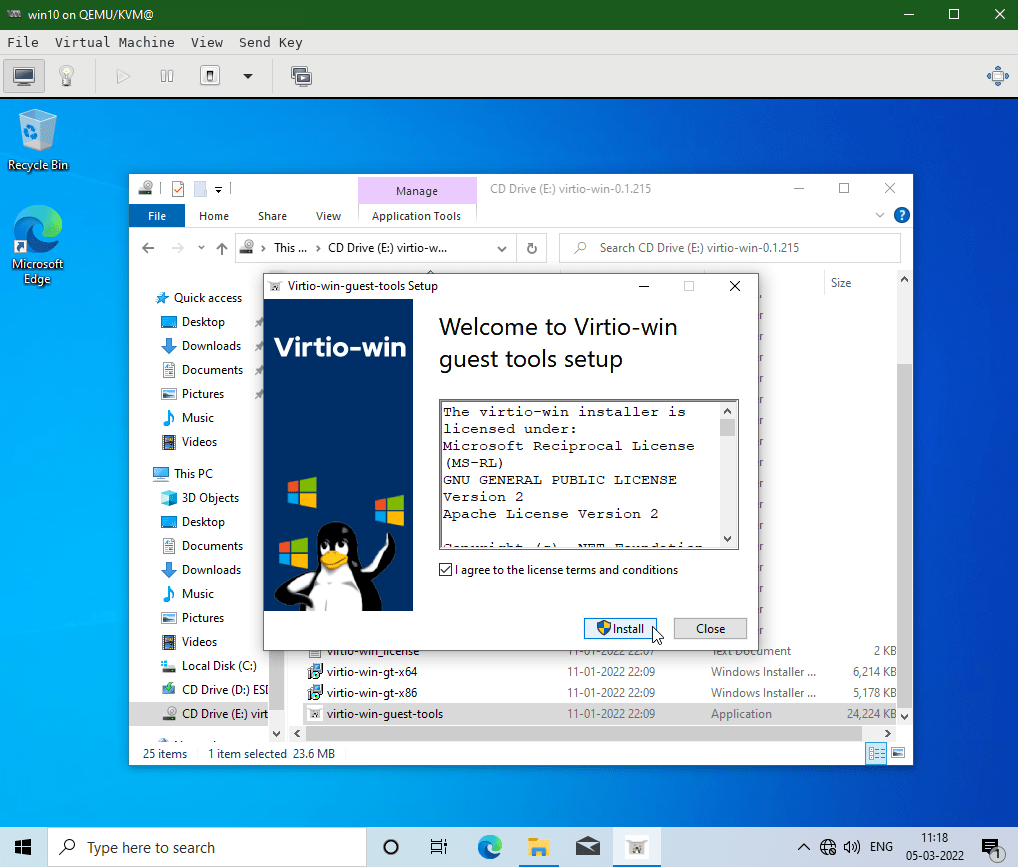




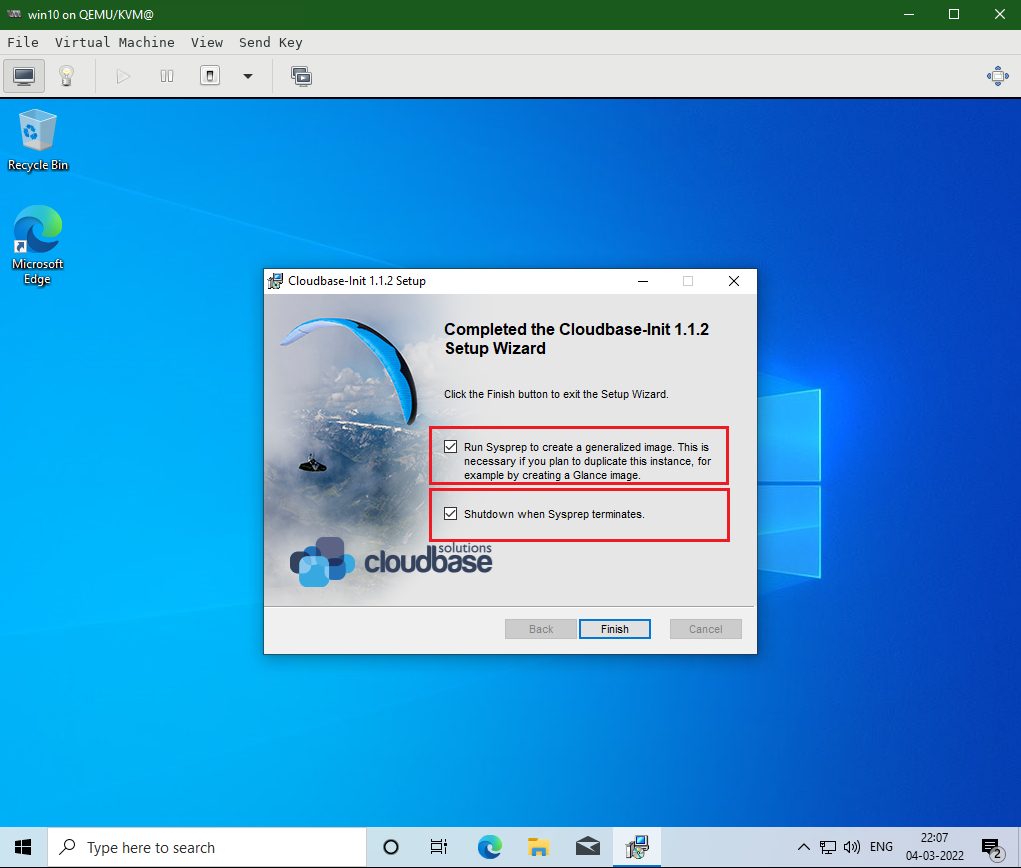


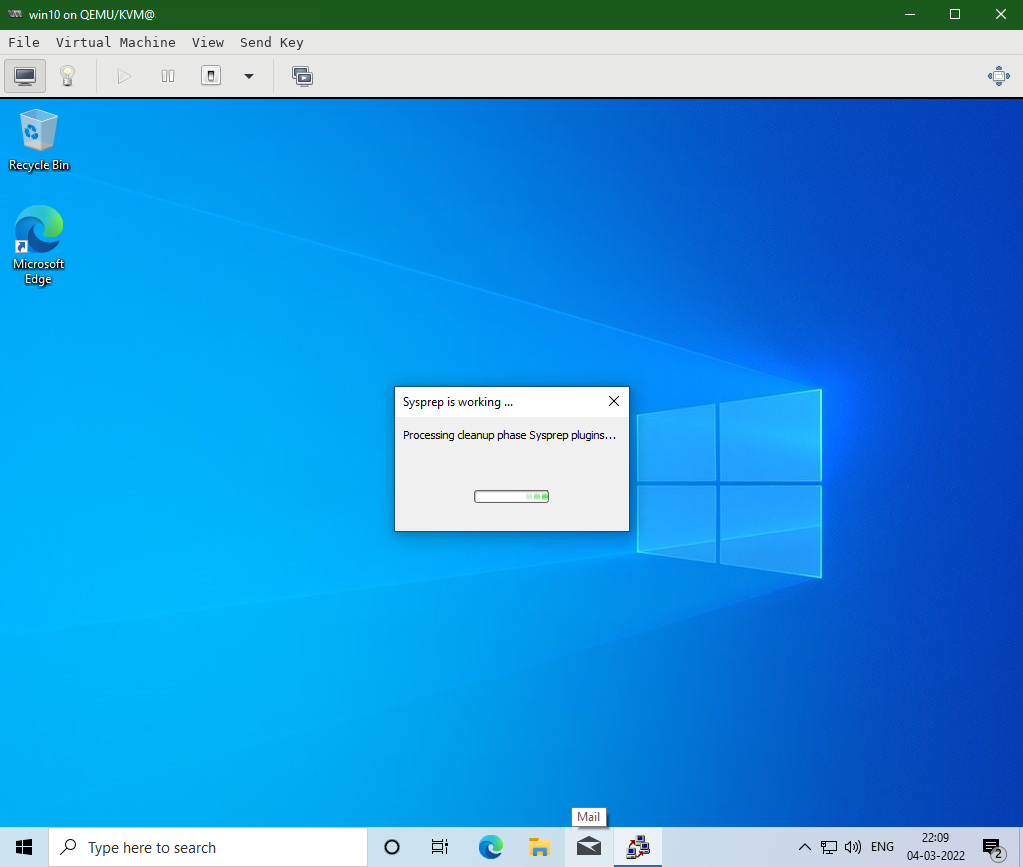






Install the Cloudbase for windows X64





# 6. How to login to our OpenStack platform

<http://172.26.13.41/auth/login/?next=/> (admin/adminpassword)