Install Kubernetes on Openstack VMs

Table of Contents

Steps detailing how to install kubernetes on VMs provisioned by openstack.

Prerequisite

We shall be using vmware workstation instead of virtualbox, the reason being since virtual box does not virtualization parameters to openstack VMs, the VMs response is slow, the `kubeadm init` fails with crash-loop

Here we will be installing kubernetes manually on openstack VMs, we will not be using openstack components magnum and octivia for provisioning K8s on openstack.

We shall be using devstack instead of installation using standard individual components

Setup VM on vmware workstation

We will install `ubuntu 18.04 server` as host VM.

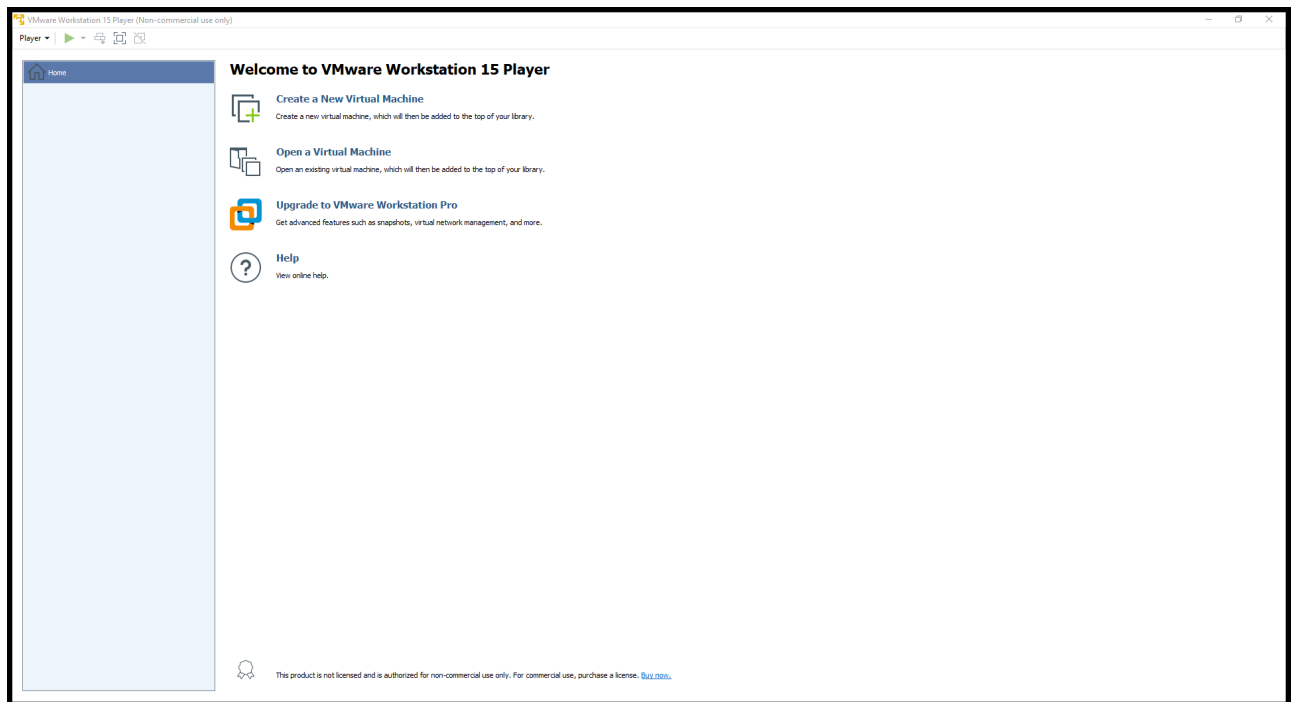1. Create new VM by clicking on `Create New Virtual Machine`

Figure 1. VM create wizard

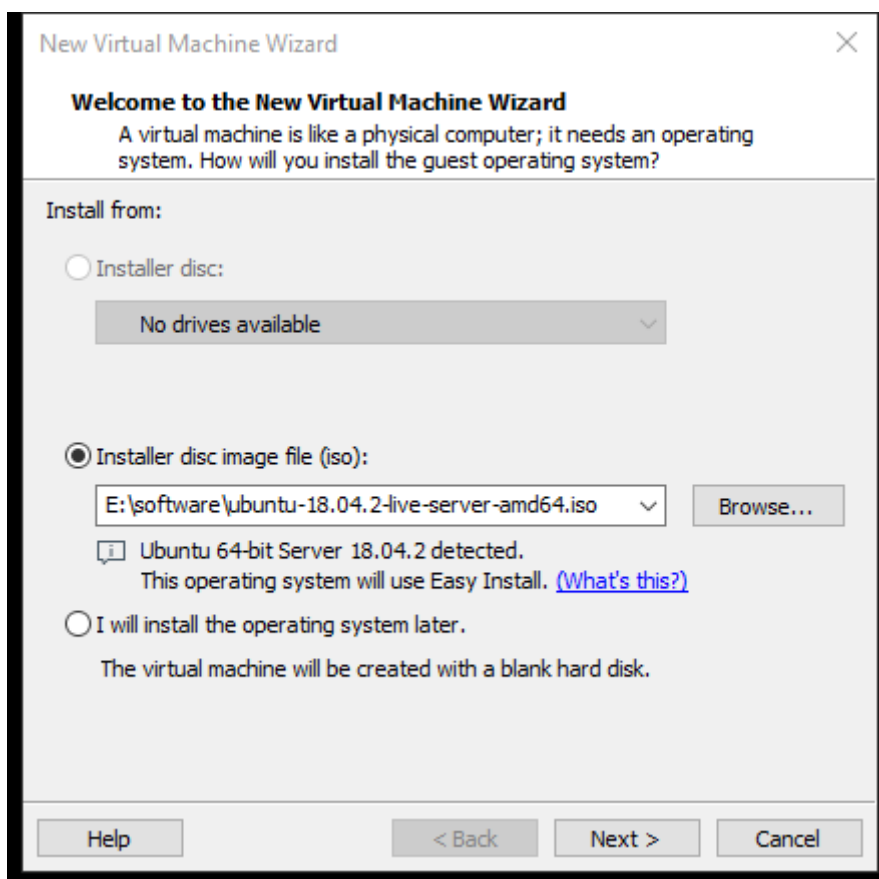2. Browse for and select `ubuntu server image` (ISO file) and click *next*
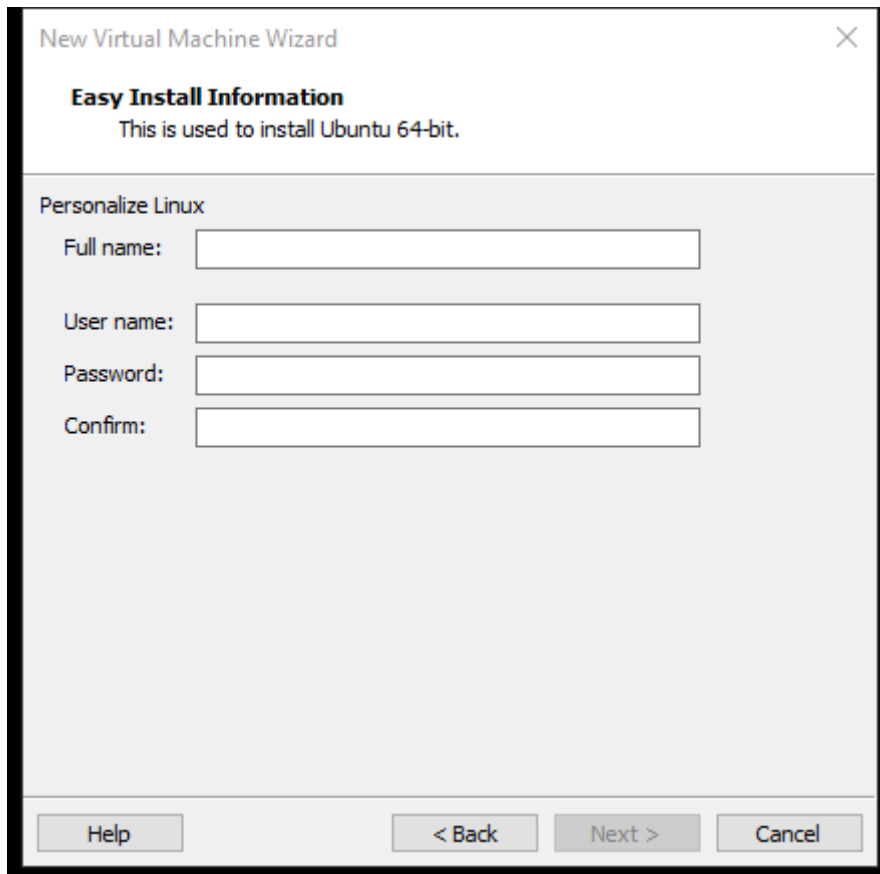
Figure 2. Select VM Image source

3. Provide VM default user information.
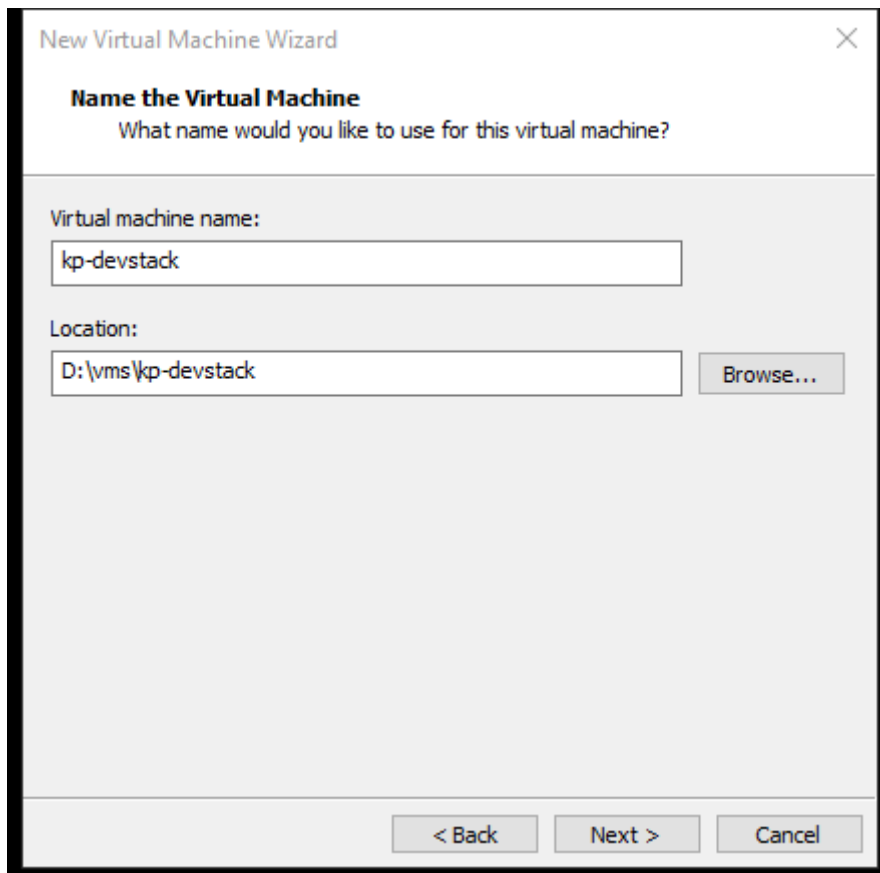
Figure 3. UserInfo wizard

4. Set VM name and disk location.

Figure 4. DiskInfo wizard

5. Set Disk size and select option `store virtual disk as a single file`
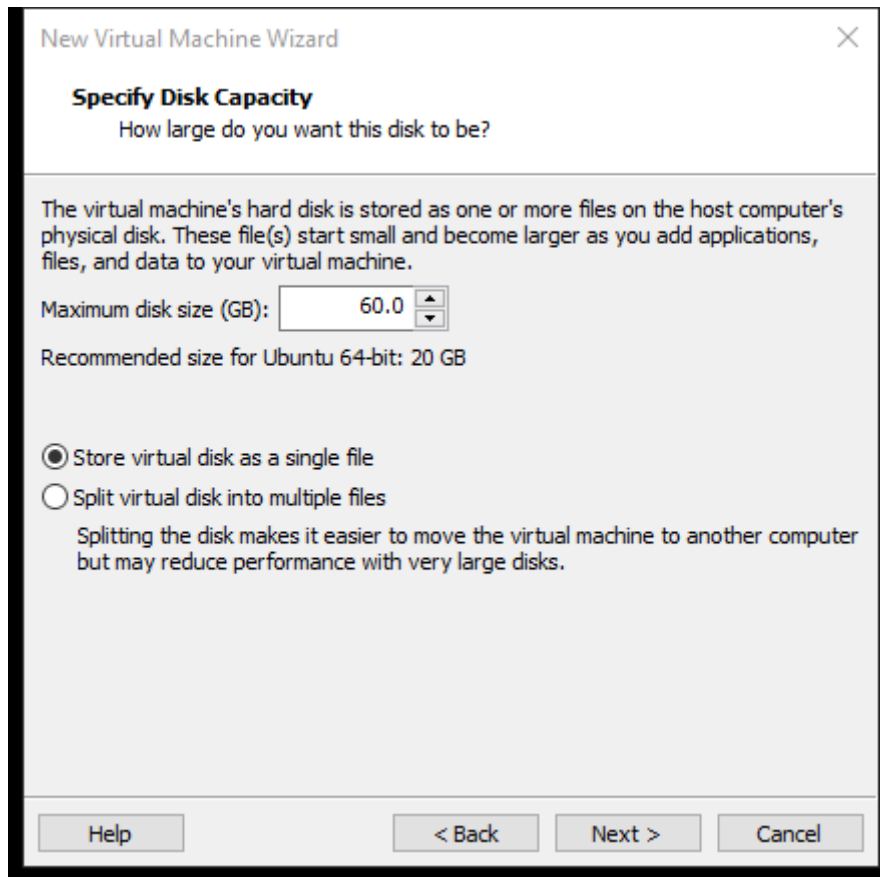
Figure 5. DiskInfo wizard

6. Click on *customize hardware* as we need to increase RAM and enable Virtualization parameters
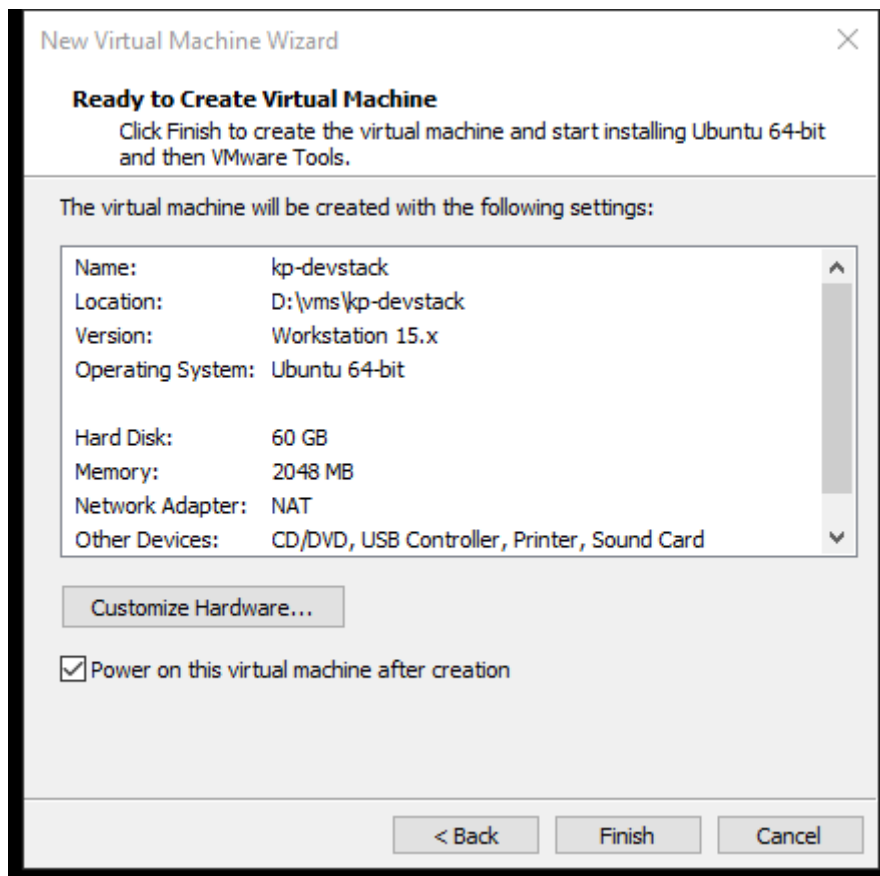
Figure 6. VM summary

7. Increase RAM size, processor count and select all options of virtualization engine.
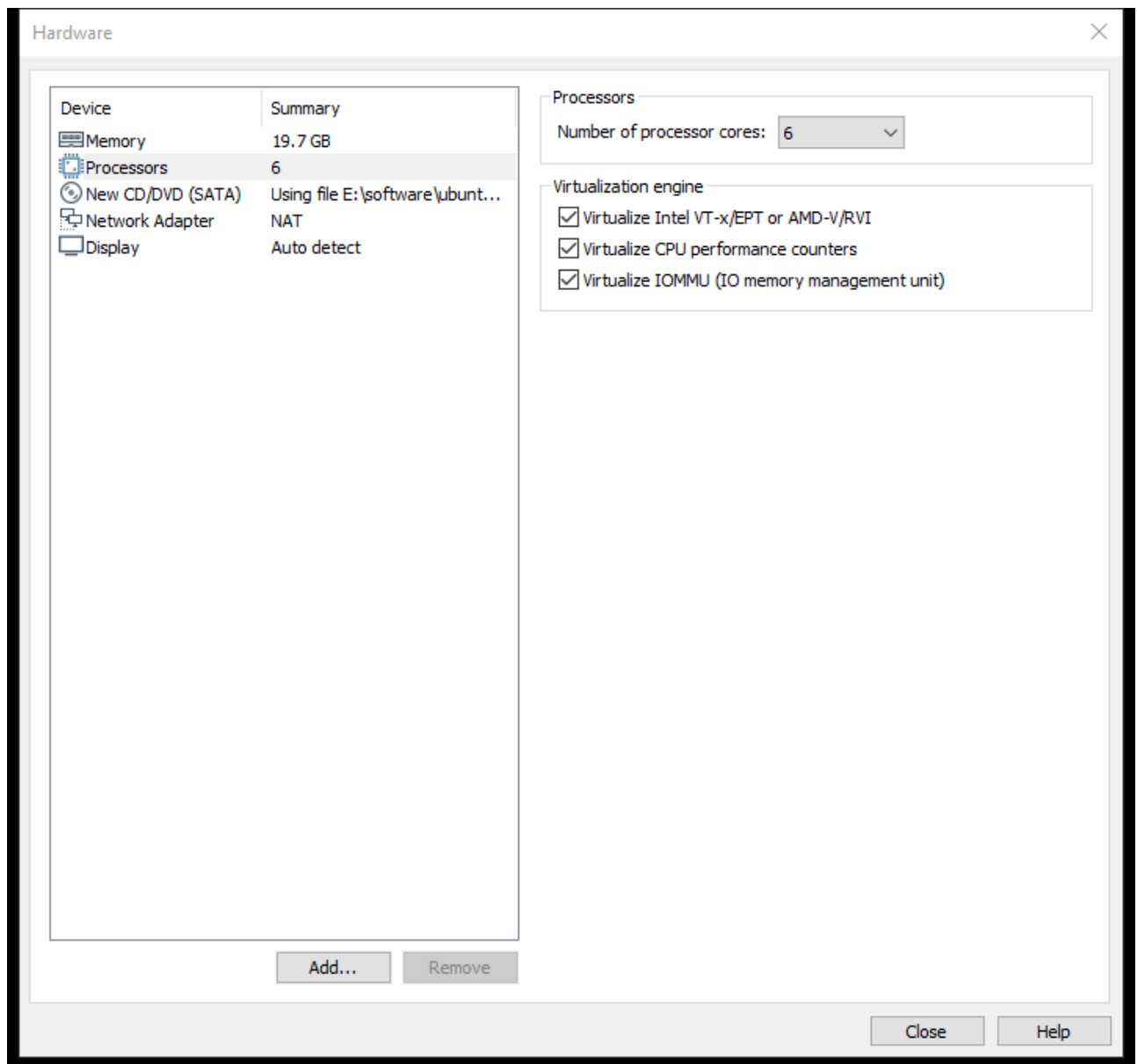
Figure 7. Hardware customization

> ⛔ You need to enable virtualization parameters for nested kvm to work.

> 💡 You can remove controllers such as usb, sound, printer and remove 3D support in display section

8. Add two additional network adapters, configure `network connection types` as shown in below table.

Name	Connection Type
Network Adapter	Host only
Network Adapter 2	NAT
Network Adapter 3	Bridged
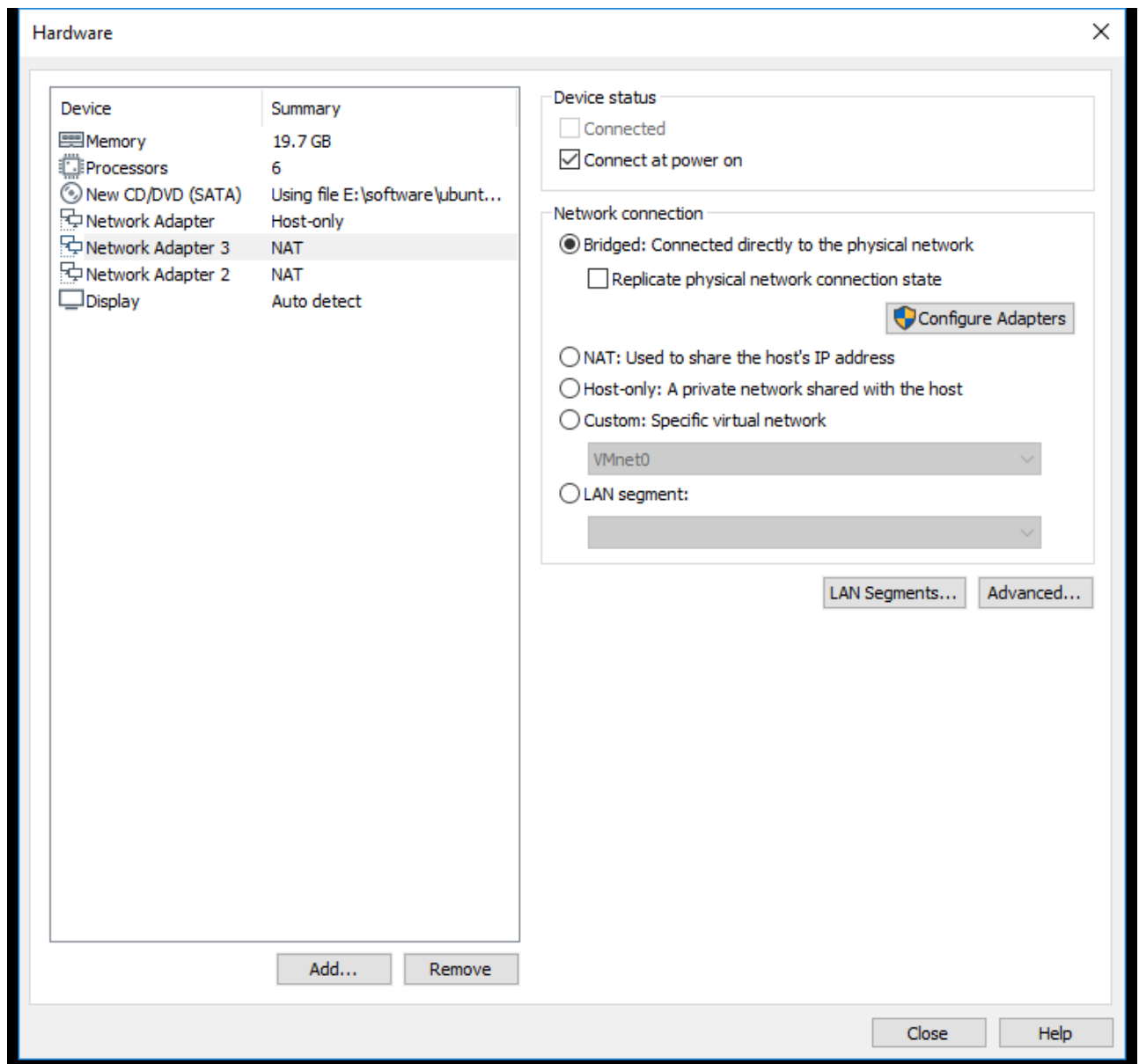
Figure 8. Network adapter settings

while configuring bridged configuration you can customize host network adapter the bridging applies to.
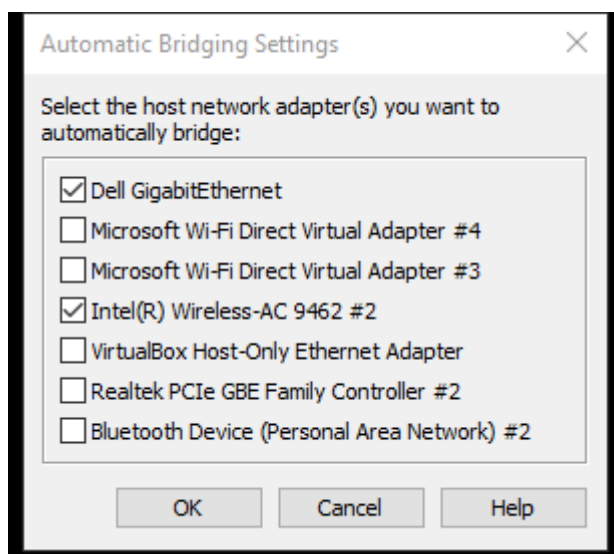
Figure 9. Bridged configuration

9. Once configuration is done, click on *close*
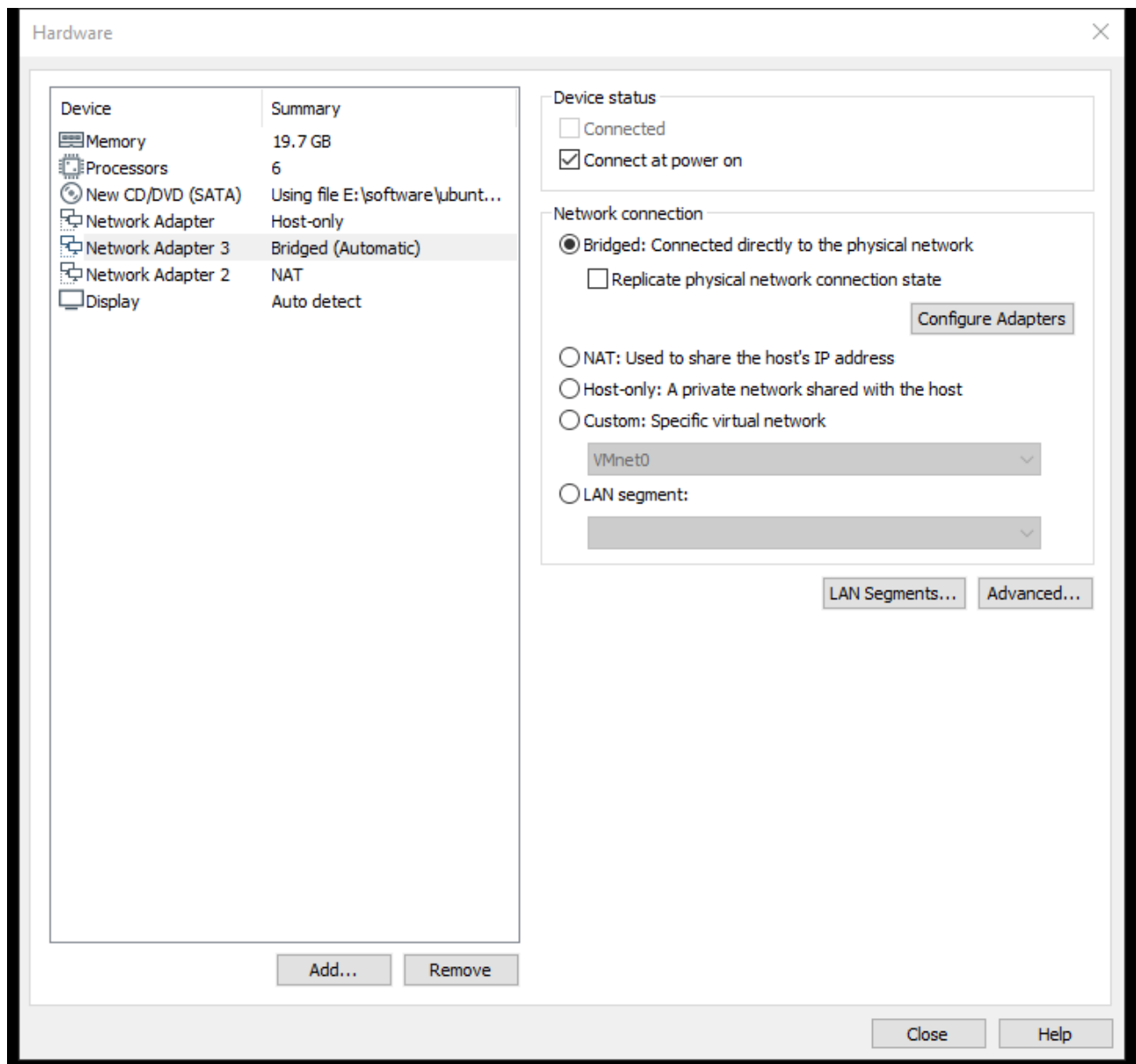
Figure 10. Final harwardware settings

10. click on *finsh*
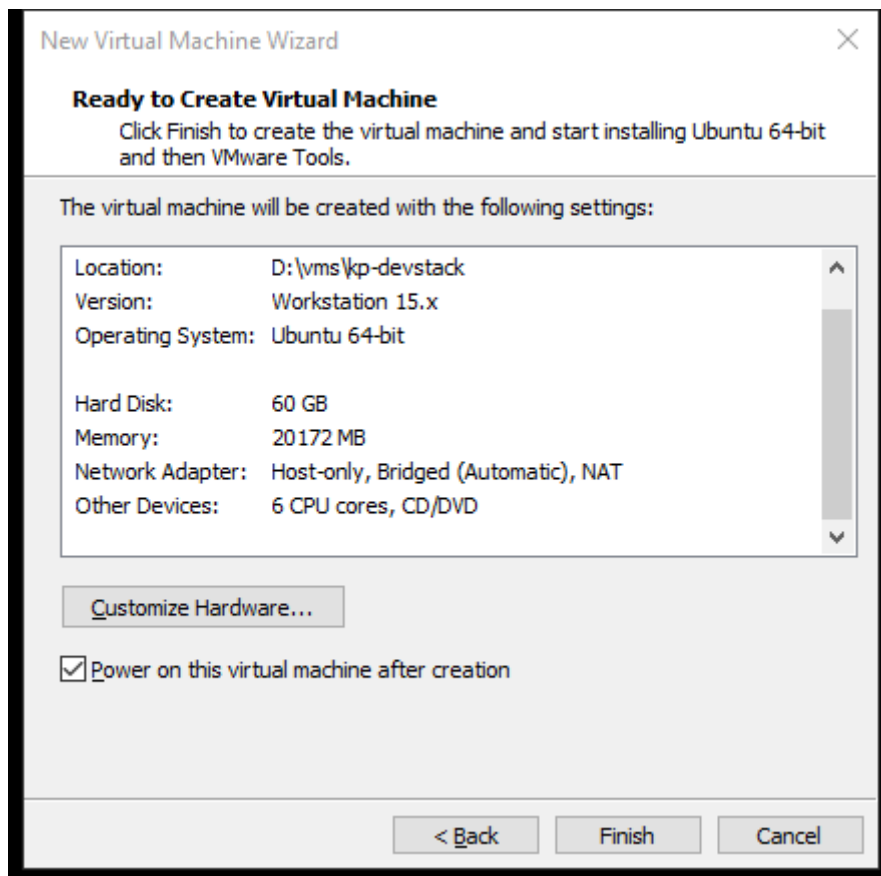
Figure 11. VM summary

on clicking of *finish* button application will setup disk space allocate hardware and will reboot appliance and we will be presented with ubuntu OS installation wizard.

Setup Ubuntu OS

1. On Image load we will be promted to select languare, set the desired language.
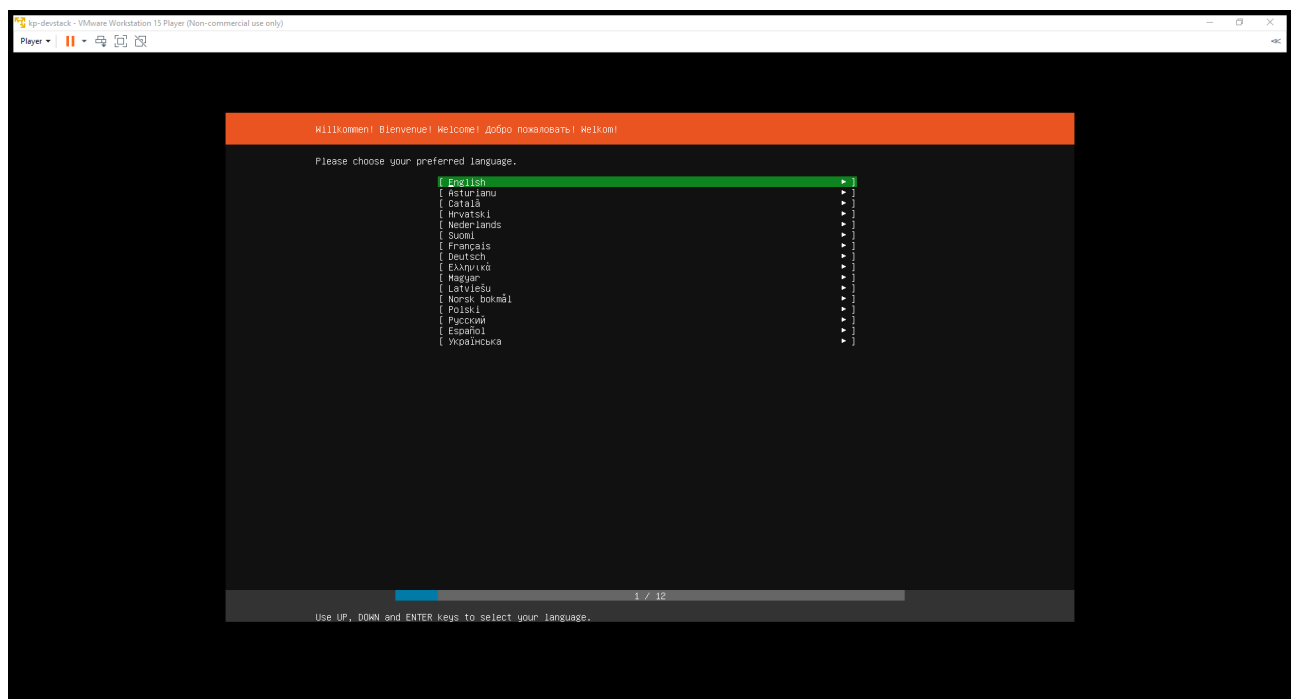
Figure 12. os setup language wizard

2. Next we will be prompted to select keyboard layout, let the default configuration be as it is. Navigate to *Done* using tab and press *enter*.
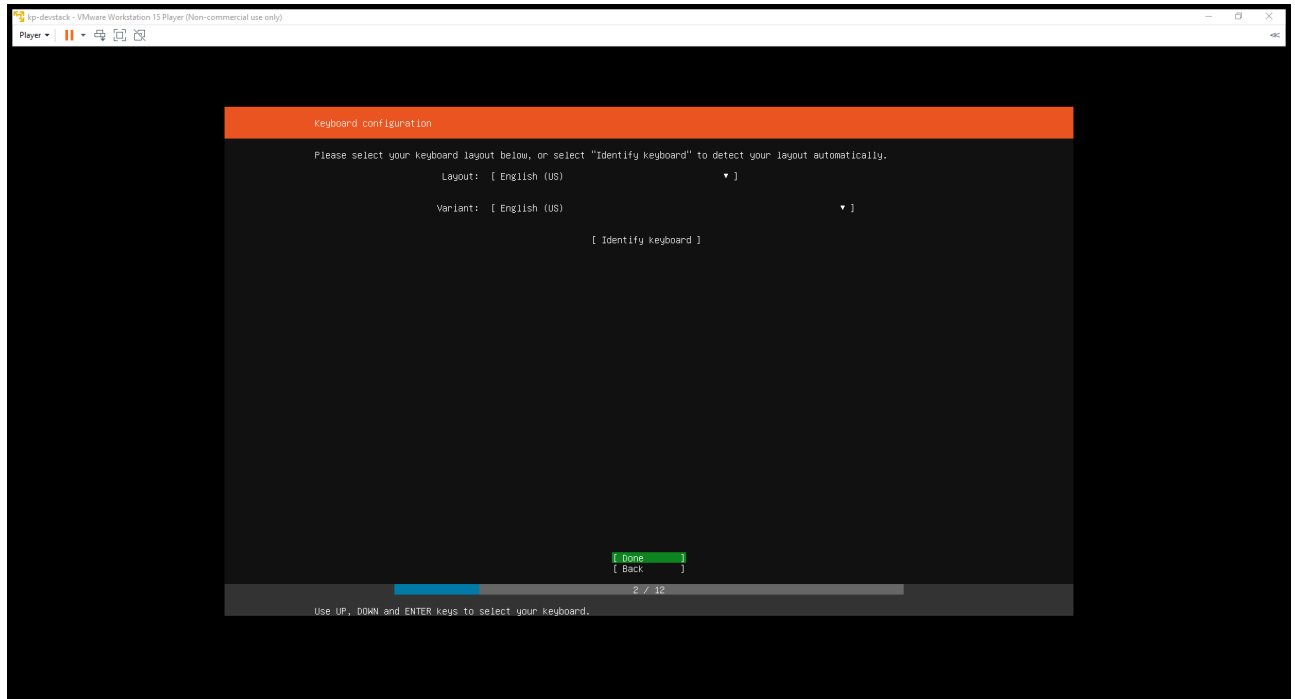
Figure 13. os setup keyboard layout wizard

3. Next we will be navigated to network interface wizard. navigate by pressing arrow button to select a interface, once interface is highlighted press enter and a drop down menu list appears, configure network settings as shown in below table.

Interface	Settings
ens32	we need set static ip to this interface, select manual option from drop down and enter static ip address from subnet, you can get the subnet from the dhcp ip already assigned to this interface (shown in second diagram below)
ens33	interface will get ip from DHCP, hence no change is required
ens34	We need to disable this interface and bring up/down manually, hence select option `disable` from option
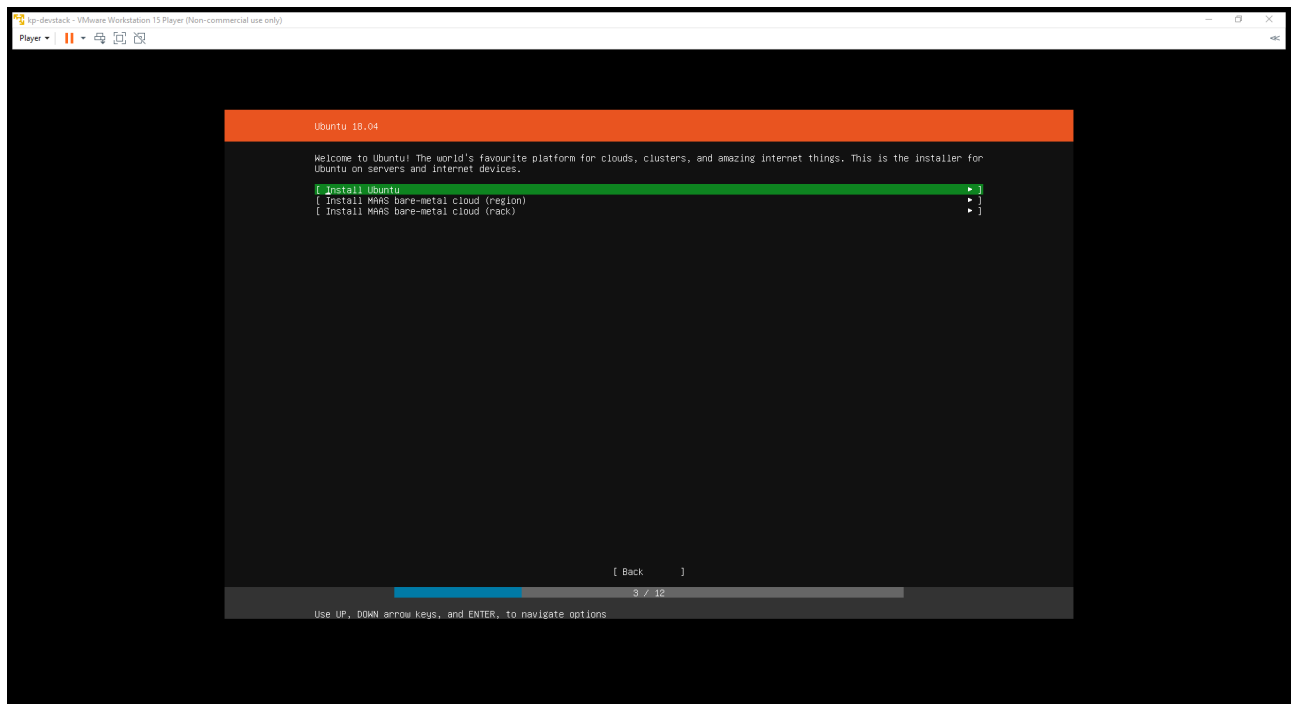
Figure 14. os setup network settings wizard
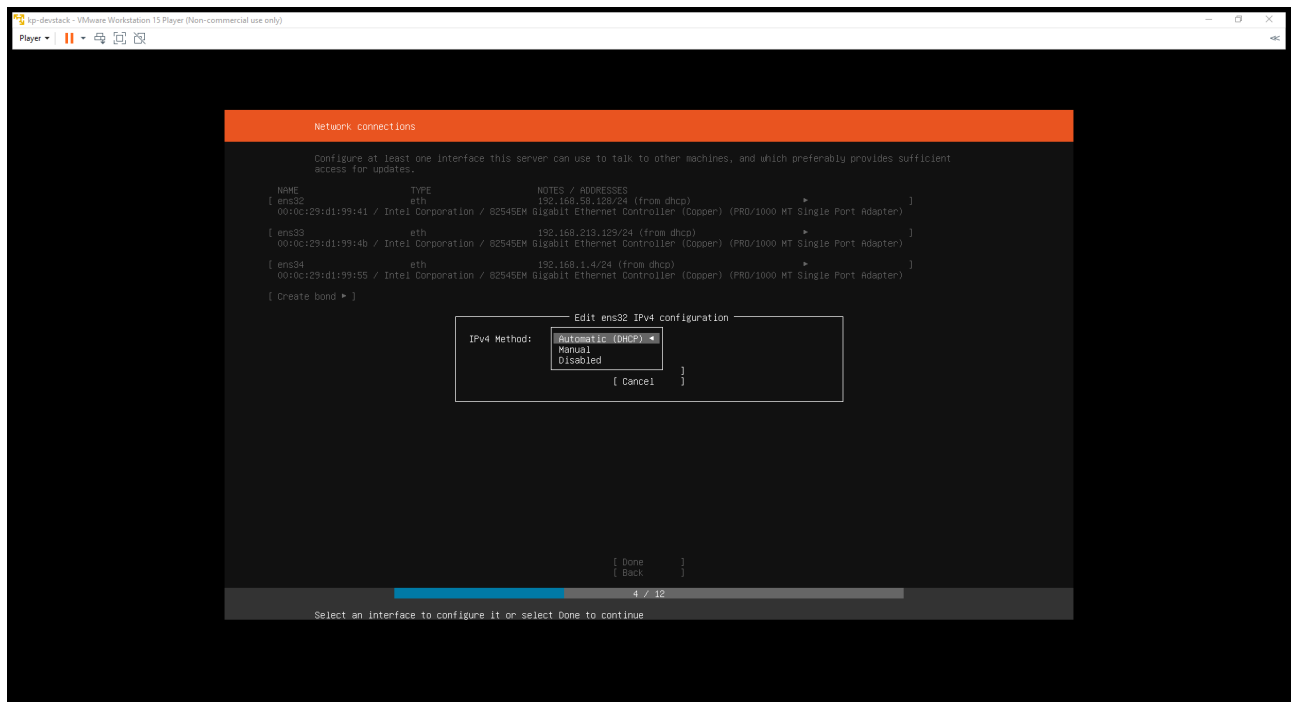
Figure 15. static ip settings wizard

4. Below is network interface summary settings, once interface settings configured, navigate to *Done* button and press *enter*
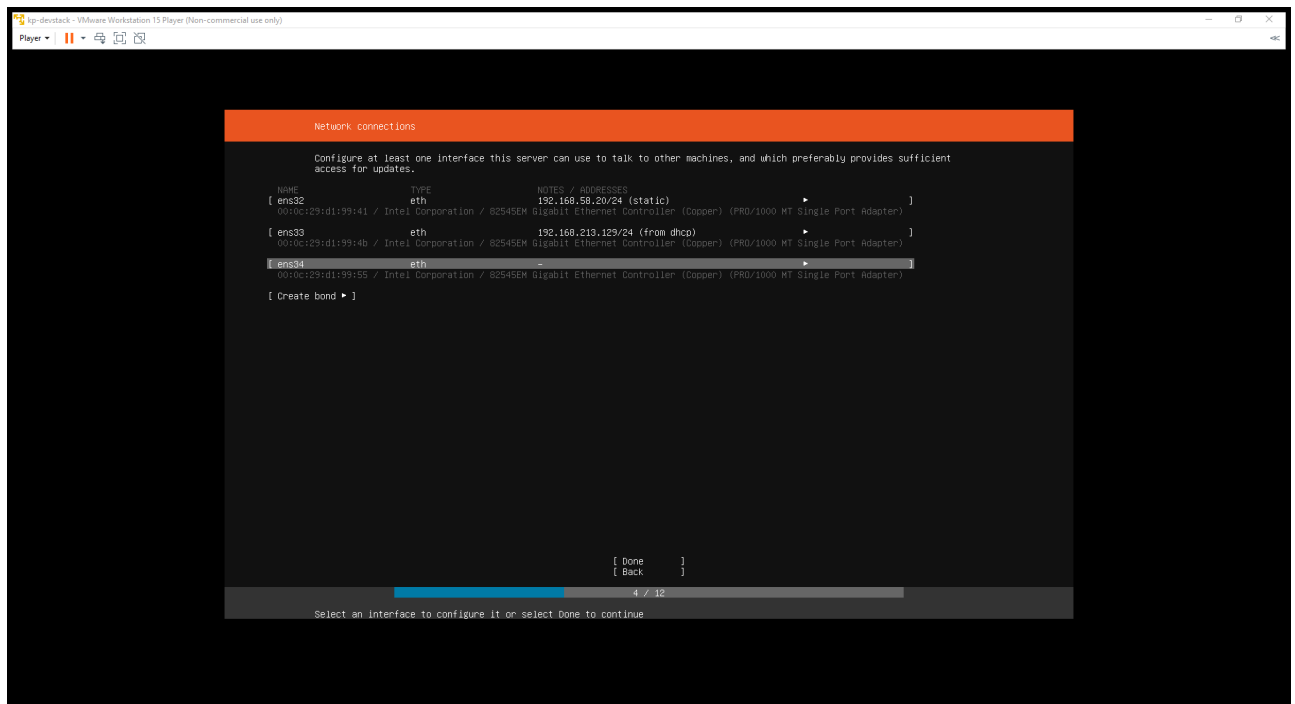
Figure 16. interface settings summary

5. Skip through proxy settings dialog by pressing enter on *done*, until unless you need to configure a proxy.
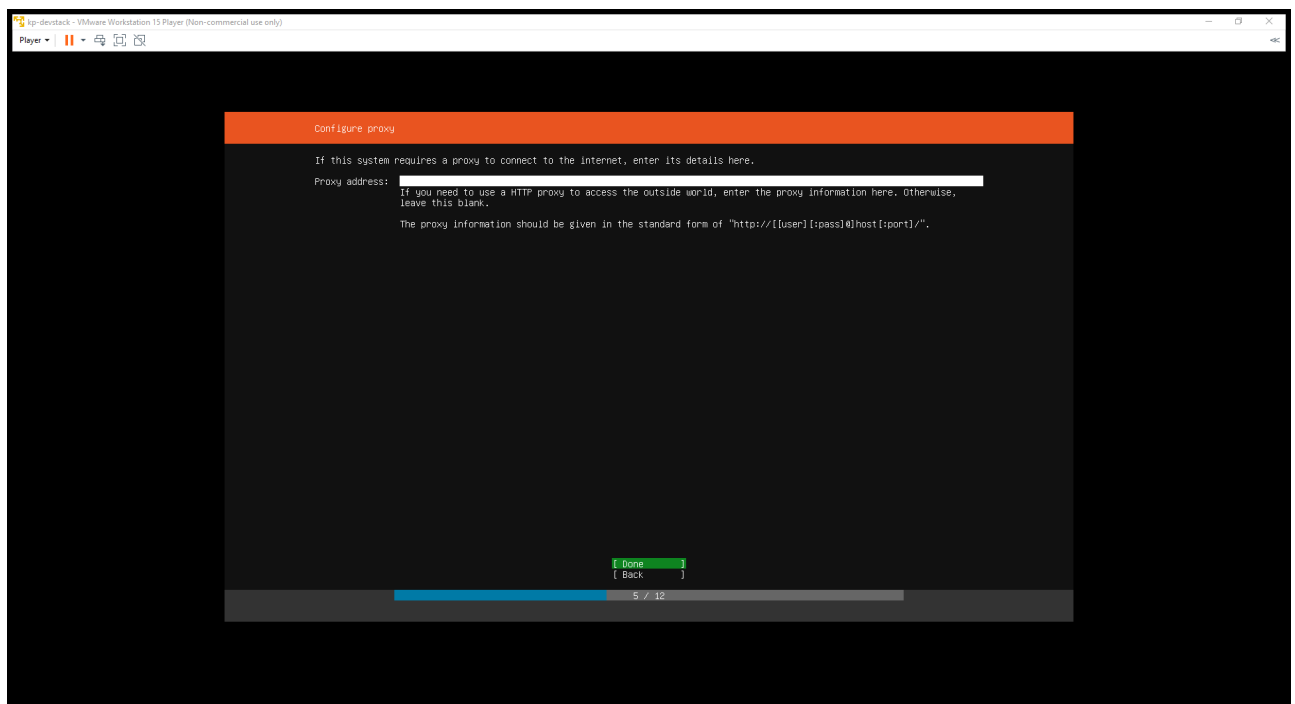
Figure 17. proxy settings wizard

6. Skip through ubuntu archive mirror page by pressing enter on *done*.
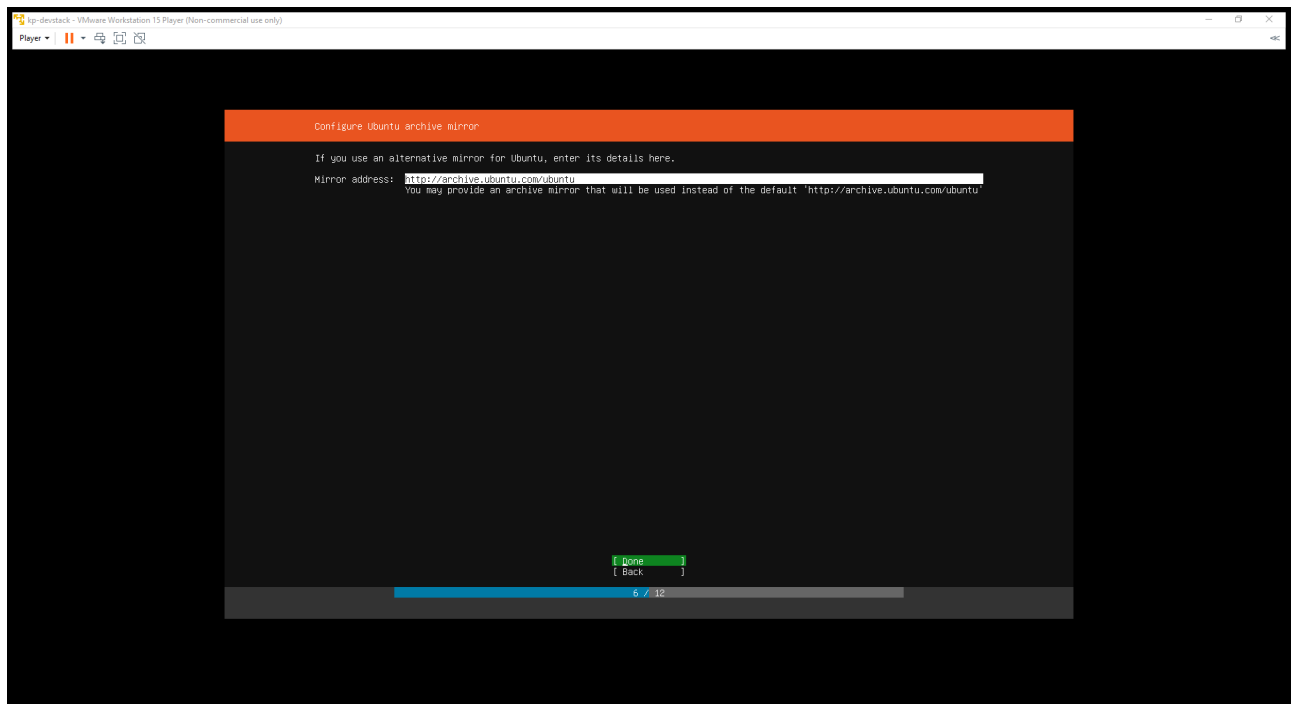
Figure 18. ubuntu archive settings wizard

7. Next we will be prompted to configure file system, select `use entire disk and with lvm` option and press *enter*.
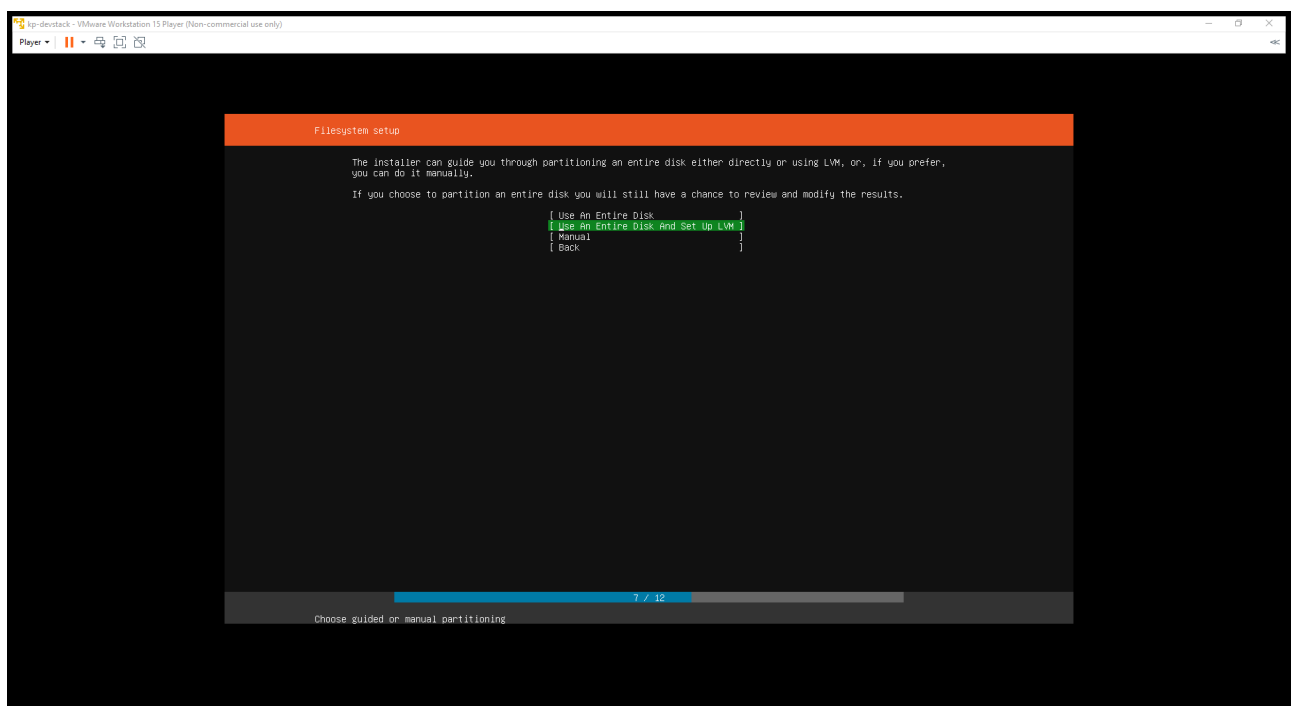
Figure 19. filesystem setup wizard

8. Since we have only one disk, file system setup wizard displays only one drive, select the drive and press *enter*
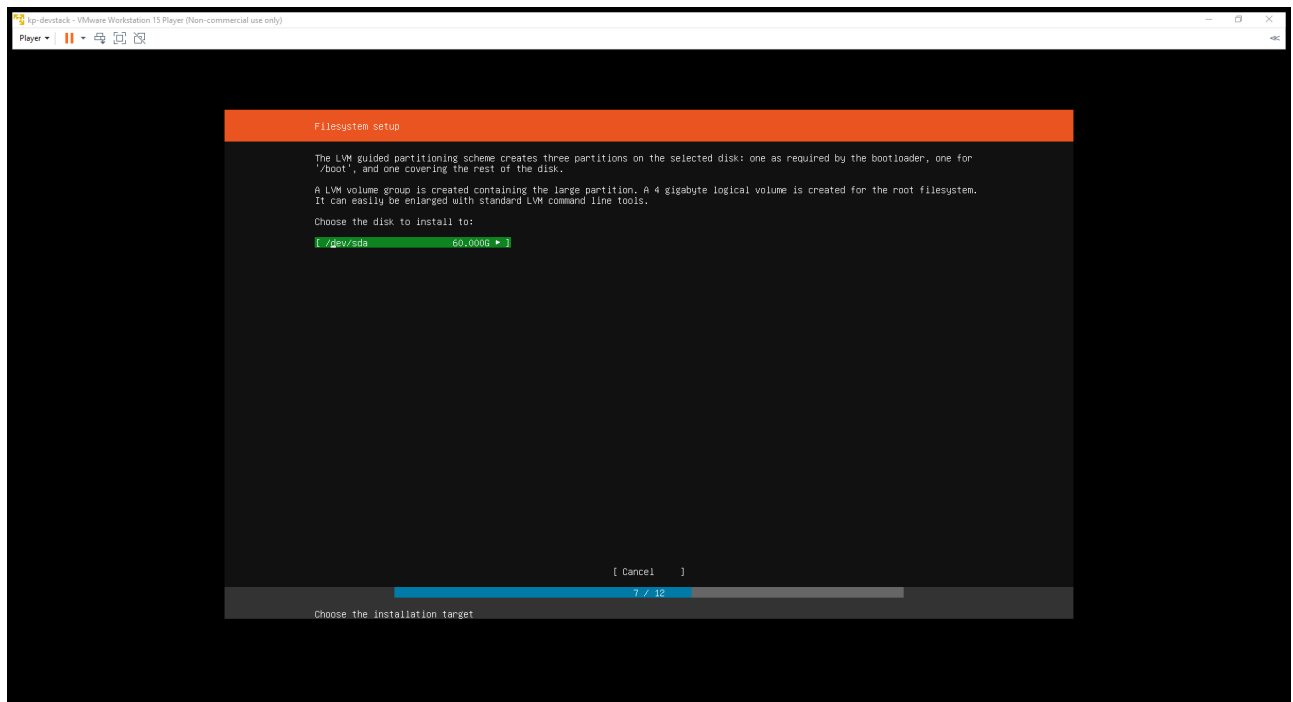
Figure 20. filesystem setup wizard

> When selecting LVM make sure you've selected added entire disk to VG by default in 18.04 4GB is selected. You can extend the drive at later point of time as well.

To set entire disk for usage navigate to `Available device` → `-vg` `-lv` by pressing *up* arrow and then press *enter*, select option `edit` from drop down and update the value to entire drive
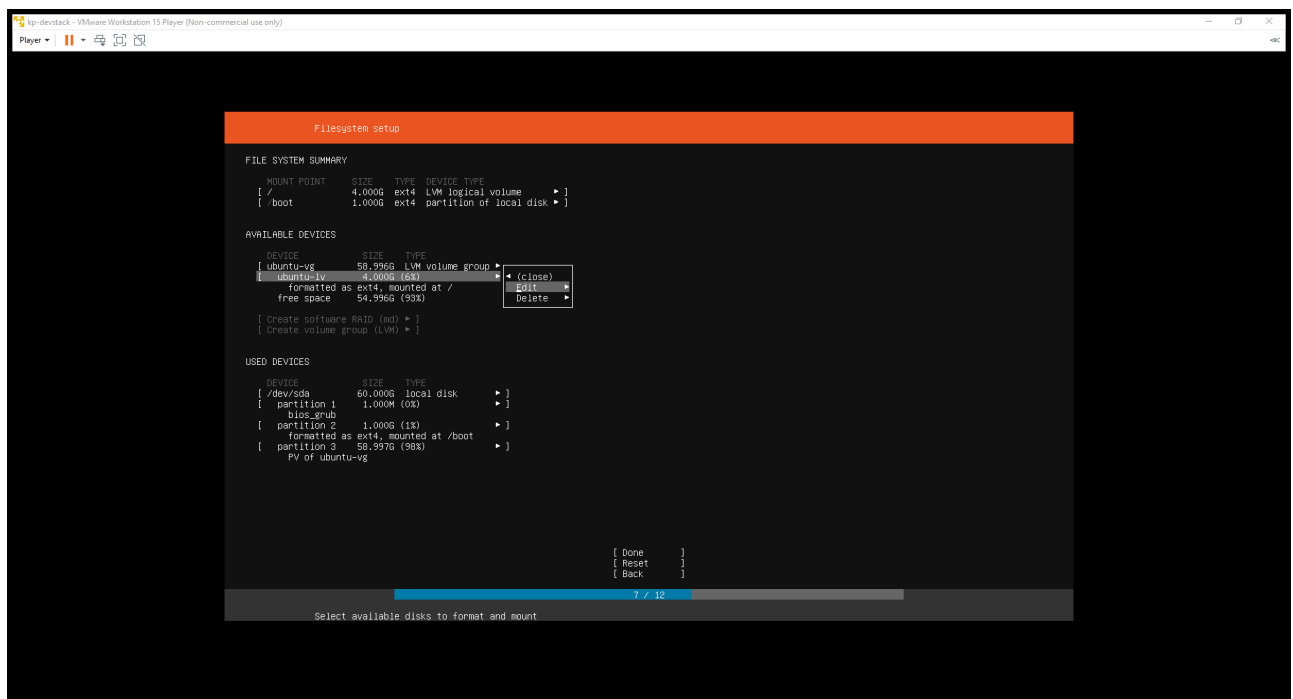
Figure 21. LVM configuration wizard
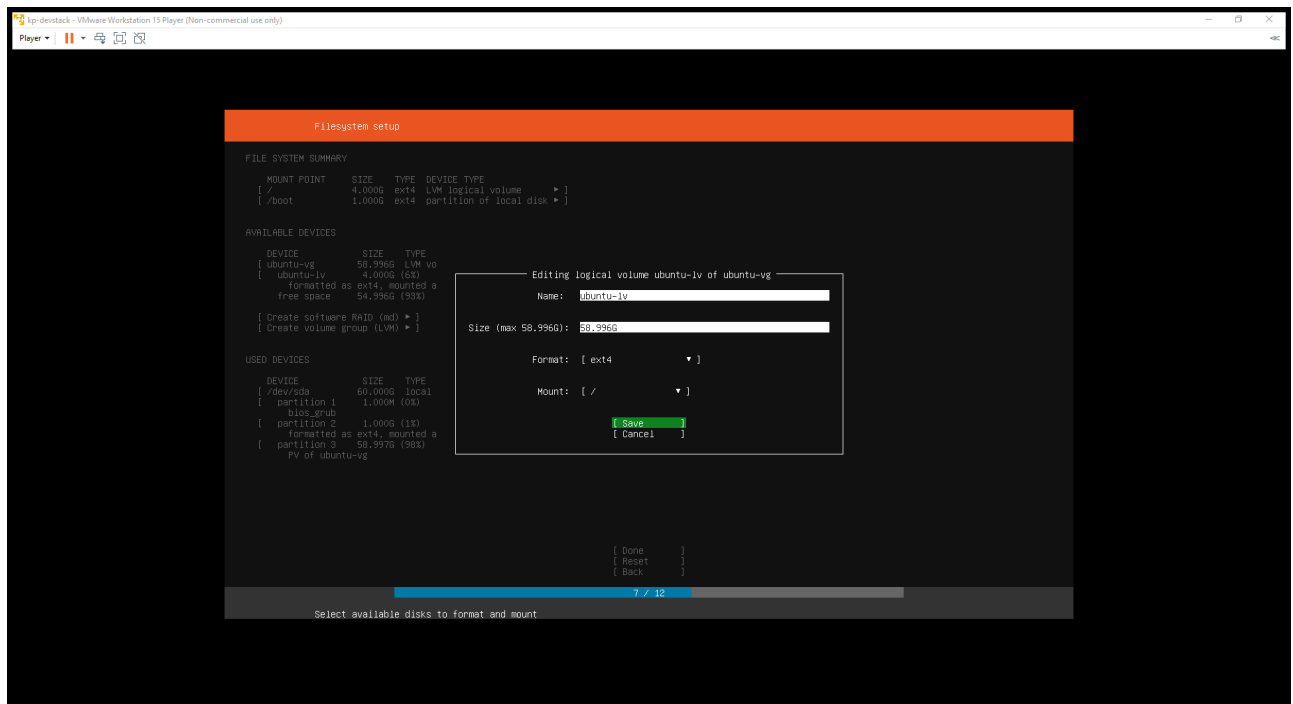
Figure 22. LVM configuration wizard

9. A summary window will shown describing the Disk Partitions, once file system configuration is done press *enter* after navigating to *done*
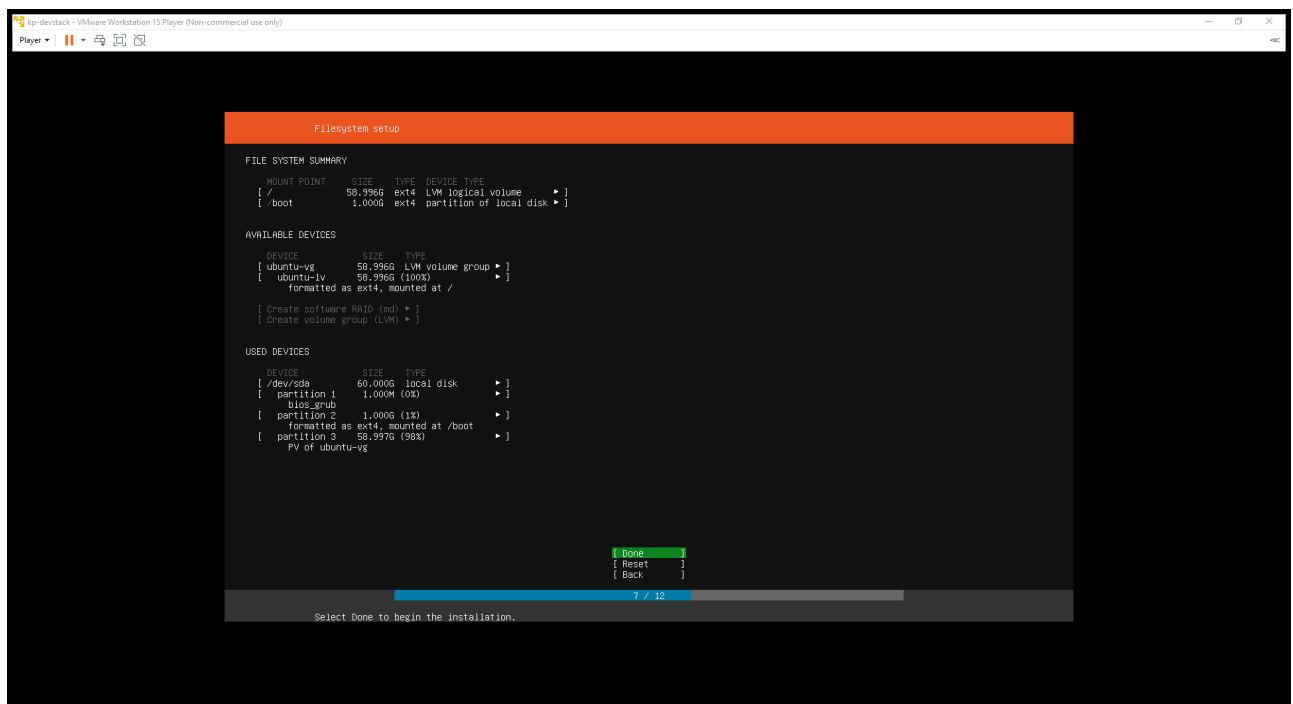
Figure 23. File system configuration summary

10. Next `Profile setup` window comes up, fill all the details.
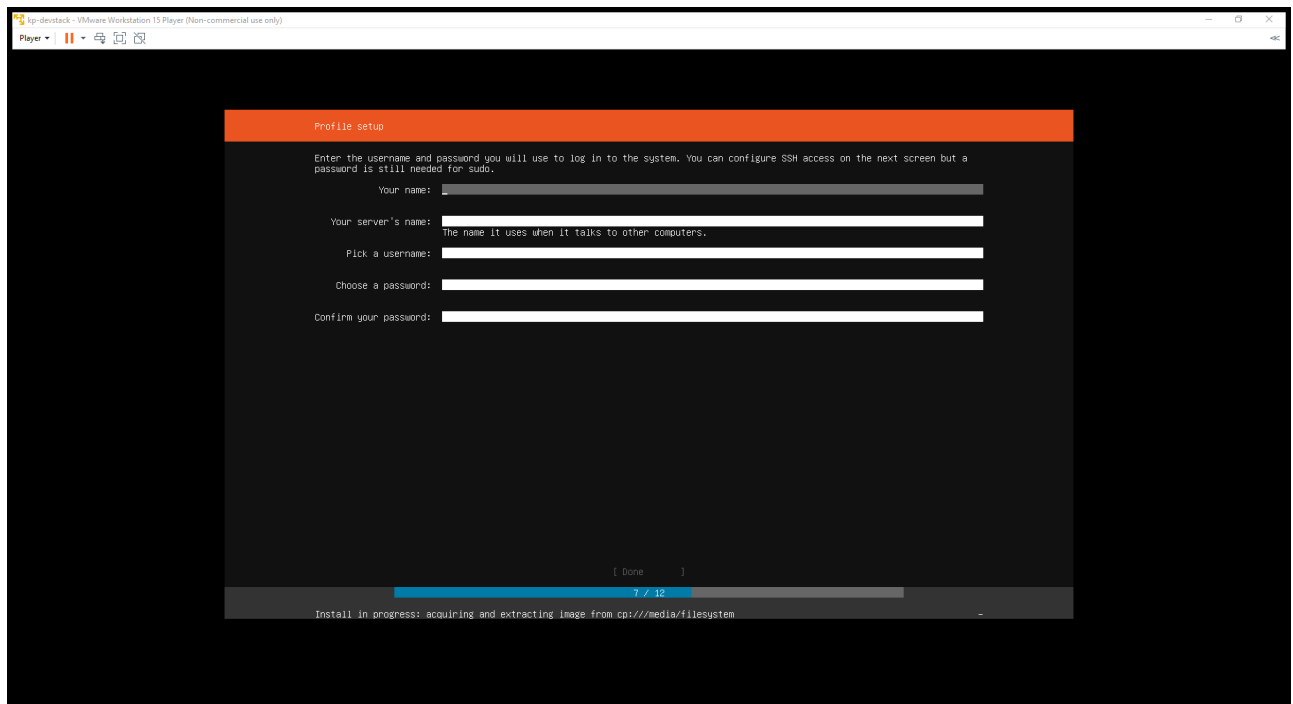
Figure 24. Profile setup wizard

11. Select option `install ssh server` before navigating to next page
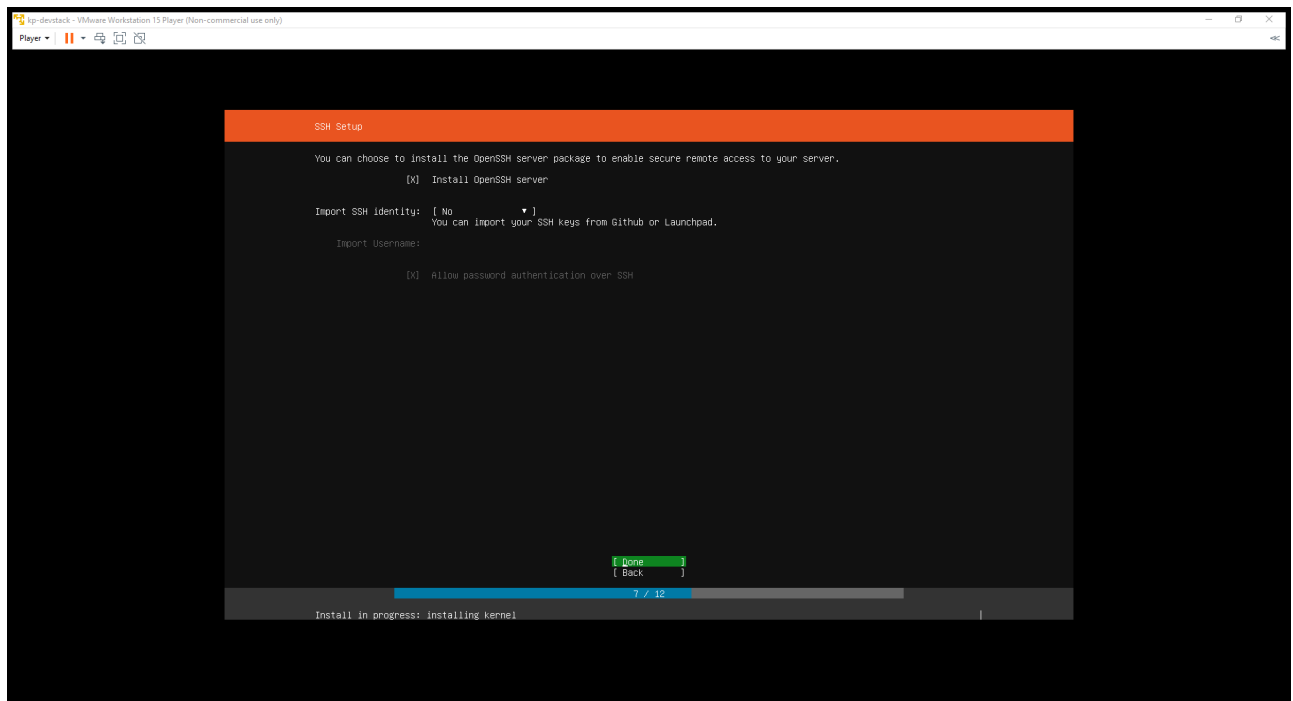
Figure 25. SSH server configuration

- Once you navigate from ssh server installation window, it would take sometime to setup OS. Once Setup completes, would prompt to install grub, select yes to install grub, finally after installation click on `Reboot`
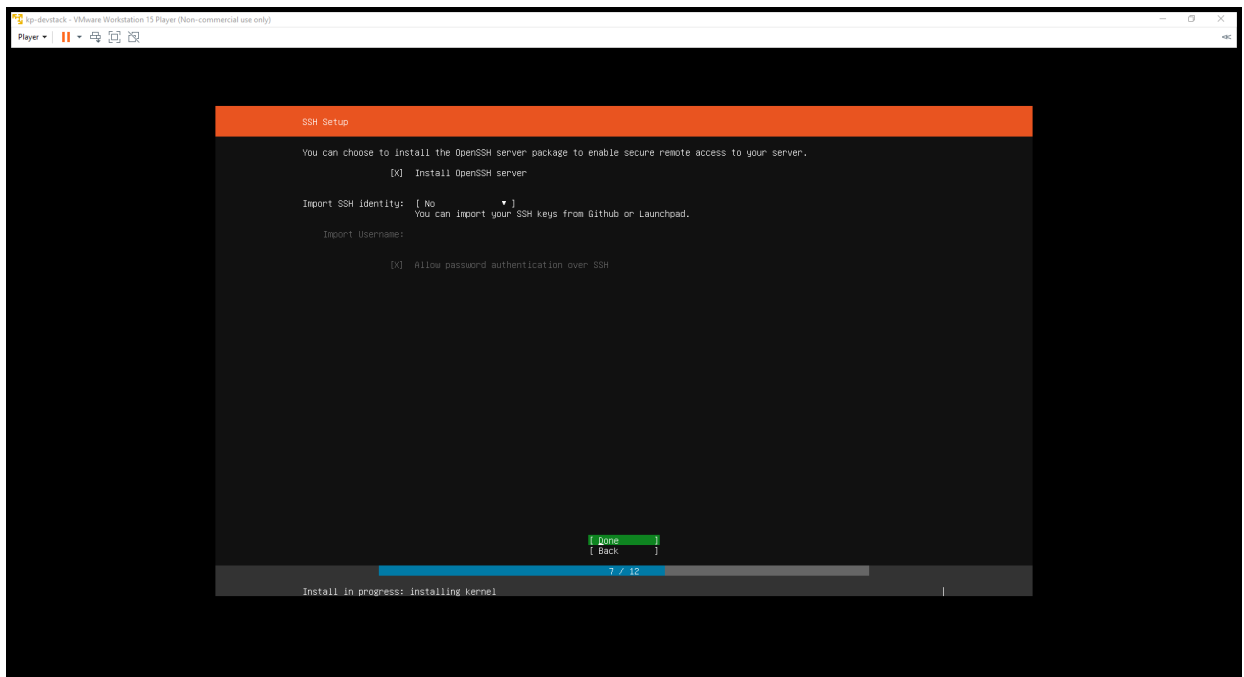
Figure 26. Installation summary

Configure bridge interface

We need to configure bridge interface to come up/down with no ip address.

1. Execute below command and note down the interface name that maps to bridge network.

```
$ ifconfig -a
```

2. Create a new file `/etc/systemd/system/manual-iface.service` and copy the below configuration

```
[Unit]
Description=Service to bring up/down unconfigured nic ens34 ①
After=network.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/sbin/ip link set ens34 up ①
ExecStop=/sbin/ip link set ens34 down ①

[Install]
WantedBy=multi-user.target
```

① Replace the interface with bridge interface

3. Enable the service.

```
$ sudo systemctl enable manual-iface.service
```

4. Start the service.

```
$ sudo systemctl start manual-iface.service
```

5. Now executing below command shall list the interface.

```
$ ifconfig
```

Install devstack

We will be using stein release(which is the last stable release while writing this documentation)

1. Change directory to /opt

```
$ cd /opt
```

2. clone devstack git repository

```
$ sudo git clone https://github.com/openstack/devstack.git -b stable/stein
```

3. change directory ownership

```
$ sudo chown ${USER}:${USER}
```

4. change directory to devstack

```
$ cd devstack
```

5. Download the local.conf.

```
$ wget -c
https://gist.githubusercontent.com/kprasad99/f4cfa3ef7e2548685c9f7e214046f071/raw/f
b2b4b2c1fe3efa060d98025c40285bf9e2bc913/local.k8s.conf \
      -O local.conf
```

> **!** Replace values of HOST_IP and FLAT_INTERFACE according to your system settings

6. Follow the steps described here and validate `nested-kvm` is enabled

7. Execute installation script.

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

Installation process will take quite time first time, if fails run `./unstack.sh` and `./clean.sh` and then rerun `./stack.sh`

Once installation proccess completes, you will see output something as shown below, you can use the url provided in output to explore using UI.

```
========================
DevStack Component Timing
 (times are in seconds)
========================
run_process            43
test_with_retry         3
apt-get-update          3
osc                   127
wait_for_service       23
git_timed             334
dbsync                 76
pip_install           457
apt-get               864
------------------------
Unaccounted time     1624
========================
Total runtime        3554



This is your host IP address: 192.168.58.20
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.58.20/dashboard ①
Keystone is serving at http://192.168.58.20/identity/
The default users are: admin and demo ②
The password: openstack123 ③

WARNING:
Using lib/neutron-legacy is deprecated, and it will be removed in the future


Services are running under systemd unit files.
For more information see:
https://docs.openstack.org/devstack/latest/systemd.html
```

① Dashboard URL to explore openstack UI

② by default to user are created by devstack.

③ Both users have same password

Provision K8s VMs

We shall bring two servers on the openstack which would be part of k8s cluster

We shall install ubuntu cloud image for VMs, you can download ubuntu 18.04 cloud image from here

Admin Tasks

Below configuration need to be performed using admin user.

1. Load admin credentials.

```
$ source /opt/devstack/openrc admin
```

2. Create a custom flavour

```
$ openstack flavor create --ram 7168 --disk 15 --vcpus 2 7G15G
```

> instead of creating custom flavour, you can run below command and use one of the flavor already configured by devstack
>
> ```
> $ openstack flavor list
> ```

3. Upload ubuntu.18.04 cloud image to glance.

```
$ openstack image create --disk-format qcow2 \
    --file images/bionic-server-cloudimg-amd64.img \
    --min-disk 10 \
    --min-ram 4096 \
    --public ubuntu.18.04
```

> change the file path to the location of ubuntu cloud image download location

4. Allocate two floating ips to project demo by executing below command twice.

```
$ openstack floating ip create public --project demo
```

Project Task

We will be creating k8s cluster under project demo.

1. Load demo credentials.

   ```
   $ source /opt/devstack/openrc demo
   ```

2. (Optional) Create directory to store pem file.

   ```
   $ mkdir vm_ssh_keys
   ```

3. Create ssh keypair, so that we can login to VM, by default ubuntu cloud image doesn't ship with passwrod login, we need to use ssh first time to login VM.

   ```
   $ openstack keypair create k-ssh > vm_ssh_keys/k8s.priv
   ```

4. Change file permission else we would get login permission denied error.

   ```
   $ chmod 400 vm_ssh_keys/k8s.priv
   ```

5. Execute below set of commands to create custom security group which enable ssh and icmp(optional) ports from external source.

   ```
   openstack security group create k-sg
   openstack security group rule create --protocol icmp --egress k-sg
   openstack security group rule create --protocol icmp --ingress k-sg
   openstack security group rule create --protocol tcp --dst-port 22 --ingress k-sg
   ```

6. Now create two VMs. namely k8s-master and k8s-s1

   ```
   $ openstack server create --image ubuntu.18.04 \
       --flavor 7G15G --network private \
       --security-group k-sg  \
       --key-name k-ssh k8s-master
   $ openstack server create --image ubuntu.18.04 \
       --flavor 7G15G --network private \
       --security-group k-sg  \
       --key-name k-ssh k8s-s1
   ```

7. You can check the server status be executing below command

```
$ openstack server list

//or

$ openstack server show k8s-master
```

8. Assign floating ip to VM.

   a. Execute below command and note down the private ip.

   ```
   $ openstack server list
   ```

   b. Execute below command and note down the port ids corresponding to private ip

   ```
   $ openstack port list
   ```

   c. Execute below command to list floating ip.

   ```
   $ openstack floating ip list
   ```

   d. Now use port id to be associate the floating ip

   ```
   $ openstack floating ip set 192.168.1.246 \
       --port 41c4bb1c-d146-4d5c-91a1-db70856d1820
   ```

   Repeat above step to assign floating ip to other machine as well.

9. SSH to VM.

   ```
   $ ssh -i vm_ssh_keys/k8s.priv ubuntu@192.168.1.246
   ```

Install and configure Kubernetes cluster

- Update Packages.
  - Update apt sources.

    ```
    $ sudo apt update
    ```

  - Install packages

```
$ sudo apt upgrade
```

Install Docker

- Install utility packages

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

- Add GPG key

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- Add docker repository to apt sources.

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

- Update apt sources and install docker

```
$ sudo apt update ; sudo apt install docker-ce
```

- Verify docker service is started.

```
$ sudo systemctl status docker
```

Prepare server

ⓘ | You can skip step 6.

Execute the below steps on both the VMs.

1. Add Kubernetes source gpg.

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

2. Next add kubernetes repository

```
$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

Browse http://apt.kubernetes.io/ and search for the corresponding ubuntu version, since there was no kubernetes-bionic, installing kubernetes-xenial in my case.

3. Install `kubeadm`

```
$ sudo apt update; sudo apt install kubeadm kubelet kubectl
```

4. Turn off swap

```
$ sudo swapoff -a
```

5. Turn off swap permanently in `/etc/fstab` by executing belo command.

```
$ sudo sed -i '/swap/s/^\(.*\)$/#\1/g' /etc/fstab
```

If 4 and 5 steps not followed, kubelet service will not start.

6. Add private ip to kubernetes arguments

While installing `kubeadm` you might get error related to port forwarding if you are using private IP which is true in our case. To resolve this issue, Create file `/etc/default/kubelet` and following content in the file.

```
KUBELET_EXTRA_ARGS=--node-ip=192.168.56.10
```

7. Use systemd for cgroups.

Create file `/etc/docker/daemon.json` and add following content.

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

8. Reload configuration and restart docker service.

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

9. Disable port security for communication within subnet.

```
$ openstack security group rule create --protocol any \
    --remote-ip 10.11.12.0/24 --ingress k-sg
```

10. Configure node port to be accessed from external network.

```
$ openstack security group rule create --protocol tcp
    --dst-port 30000:32767  --ingress k-sg
```

Configure Master VM.

1. Initialize kubernetes master node by executing below command.

```
$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

2. As output of above command shows to execute below command, run below command to update auth details for kubectl command.

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

3. Note(copy to textpad) the kubeadm join command which needs to be executed on slave nodes to join to the cluster

4. Install pod network.

```
$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

5. Verify all necessary pods are started

```
$ kubectl get pods --all-namespaces
```

```
NAMESPACE      NAME                                 READY   STATUS    RESTARTS   AGE
kube-system    coredns-86c58d9df4-c68gd             1/1     Running   0          6m41s
kube-system    coredns-86c58d9df4-q5bht             1/1     Running   0          6m41s
kube-system    etcd-k8s-master                      1/1     Running   0          6m6s
kube-system    kube-apiserver-k8s-master            1/1     Running   0          5m59s
kube-system    kube-controller-manager-k8s-master   1/1     Running   0          5m56s
kube-system    kube-flannel-ds-amd64-stb29          1/1     Running   0          49s
kube-system    kube-proxy-882ms                     1/1     Running   0          6m41s
kube-system    kube-scheduler-k8s-master            1/1     Running   0          5m54s
```

Configuration slave node.

1. Now go to slave node and execute the join command previously saved when you were executing kubeadm on master.

```
$ kubeadm join 192.168.56.10:6443 --token t0j1zi.v5lojsnpjh9r0rbn \
      --discovery-token-ca-cert-hash sha256:40b1142d9002003ab5b085776b8b8cba4a41ceaafab06429c49eaedc2b2939fa
```

> ℹ The above command is sample, the values are dynamically generated.

2. Now go back to master and execute the below command, you should be able to see slave node added.

```
$ kubectl get nodes
```

output:

```
NAME         STATUS   ROLES    AGE     VERSION
k8s-master   Ready    master   56m     v1.13.2
k8s-s1       Ready    <none>   2m49s   v1.13.2
```

Install Helm

Helm package manager consists of helm client and tiller component which runs as pod on k8s. We will install helm client on our master server itself.

Install helm client.

1. Install helm client using snap on master server using below command.

```
$ sudo snap install helm --classic
```

2. Verify helm client is running by executing command `helm version`, it should give version details for client and for server side should return error until we install tiller.

Install Tiller.

Before installing tiller we need to create RBAC for the tiller pod.

1. Create service account for tiller.

```
$ kubectl create serviceaccount -n kube-system tiller
```

2. Provide cluster role for the above service account.

```
$ kubectl create clusterrolebinding tiller-cluster-rule \
    --clusterrole cluster-admin \
    --serviceaccount=kube-system:tiller
```

3. Now create tiller using the above service account

```
helm init --service-account tiller
```

4. Wait until tiller pod comes up.

```
kubectl get po --all-namespaces --watch
```

5. Once tiller pod is up, run below command to verify helm is installed properly,

```
helm version
```

```
the output provide version details for both client and server.
```

Install Dashboard

1. Execute below command to install kubernetes dashboard

```
$ helm install --name kubernetes-dashboard \
    --set service.type=NodePort,rbac.clusterAdminRole=true \
    --namespace kube-system \
    --debug stable/kubernetes-dashboard
```

2. Execute below command to get the node port to access dashboard

```
kubectl get -n kube-system services kubernetes-dashboard \
    -o jsonpath="{.spec.ports[0].nodePort}"
```

3. Execute below command to get the node ip, since we have only one worker node, we can directly use k8s-s1 floating ip to access dashboard

```
kubectl get nodes -o jsonpath="{.items[0].status.addresses[0].address}"
```

4. Get token key to access ui

```
kubectl describe secret $(kubectl -n kube-system get secret | \
    grep -E 'kubernetes-dashboard-token' | \
    awk '{print $1}') -n kube-system
```

5. Access the dashboard at url `https://{floating-ip of k8s-1}:{node port}`, select token option and paste the token retrieved at above step.

> **Useful commands**
>
> - to download a chart
>
> ```
> helm fetch stable/kubernetes-dashboard --untar
> ```
>
> - to search a chart
>
> ```
> helm search dashboard
> ```
>
> - to dry run a chart from a location
>
> ```
> helm install --name kubernetes-dashboard \
>     --set service.type=NodePort --dry-run \
>     --debug kubernetes-dashboard/
> ```

Install Ingress controller

Optionally if you want to expose Any API/web application install ingress controllers such as NGINX, traefik etc.

Install Traefik Ingress controller.

1. Execute below helm command to install traefik ingress.

```
$ helm install --name treafik-ingress --namespace kube-system \
    --set dashboard.enabled=true \
    --set rbac.enabled=true stable/traefik
```

Installing NGINX ingress controller.

1. Execute below helm command to install nginx ingress.

```
$ helm install --name nginx-ingress  --namespace kube-system \
    --set rbac.create=true stable/nginx-ingress
```

References

https://github.com/helm/charts/tree/master/stable/traefik - traefik

https://github.com/helm/charts/tree/master/stable/nginx-ingress - nginx

https://kubernetes.io/docs/concepts/services-networking/ingress/ - ingress controller concepts