K21Academy
Learn Cloud from Experts

ORACLE Gold Partner | Silver Microsoft Partner | DevOps Institute Registered Education Partner

Bootstrap a Kubernetes Cluster Using Kubeadm

[Edition 15]

[Last Update 210810]

For any issues/help contact : ***support@k21academy.com***

Contents

1 Introduction 3
2 Documentation 4
2.1 Kubernetes Documentation 4
3 Pre-Requisite 5
4 K8S Cluster Setup Overview 6
4.1 Ports Required between Master & Worker Node 6
4.2 Create Master & Worker Node Linux Machines 8
5 Create 3 virtual Machines to set up Kubeadm cluster 9
5.1 Create Master Node In East Region 9
5.2 Create Worker Node One In Central US 17
5.3 Create Worker Node Two In Central US 22
5.4 Virtual Network Peering 28
6 Installing Docker, Kubeadm and Other Kubectl Packages 35
7 Configure Firewall, Update Iptables Settings, Disable SELinux and Disable SWAP 39
7.1 Configure Firewall 39
7.2 Update Iptables Settings 40
7.3 Disable SELinux 41
7.4 Disable SWAP 41
8 Kubeadm To Create And Initialise a Cluster 42
9 Using Kubeadm To Join Worker Nodes To The Cluster 43
10 (Optional) Unsubscribe Pay As you Go In Azure 47
11 TroubleShooting 50
11.1 Getting Warning while Configure cgroup driver used by kubelet on control-plane 50
11.2 Getting Error while Connecting node to Master node 50
11.3 Running kubectl get nodes and getting server localhost:8080 51
11.4 Error when Running Kubeadm init Command 52
11.5 Using Expired Token When Joining Node 52
11.6 Error when Running Kubeadm init Command 53
11.7 Unable to Ping Master node to Worker after Installing Docker 54
12 Summary 60

1 INTRODUCTION

A **Kubernetes cluster** is a set of node machines for running containerized applications. If you're running **Kubernetes**, you're running a **cluster**. At a minimum, a **cluster** contains a control plane and one or more compute machines, or nodes.

This guide Covers:

- Bootstrap (Install & Configure) a Kubernetes Cluster Using Kubeadm

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. Installing Kubeadm https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
2. Bootstrapping clusters with kubeadm

   https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/
3. Creating a single control-plane cluster with kubeadm

   https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
4. Cancel your Azure subscription

   https://docs.microsoft.com/en-us/azure/cost-management-billing/manage/cancel-azure-subscription

3 PRE-REQUISITE

Ensure that you have completed following three activity guides (or you have an Ubuntu Server)

- Create account (Trial or Paid) on Azure Cloud.

  ***Note**: Follow Activity Guide*
  Register_For_Azure_Cloud_Account_Accessing_Console_ed* from portal*

4 K8S CLUSTER SETUP OVERVIEW

4.1 Ports Required between Master & Worker Node

Ideally in production setup, communication between Master & Worker nodes should be allowed only for require ports as per screenshot given below. You can also refer to K8S documentation at https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

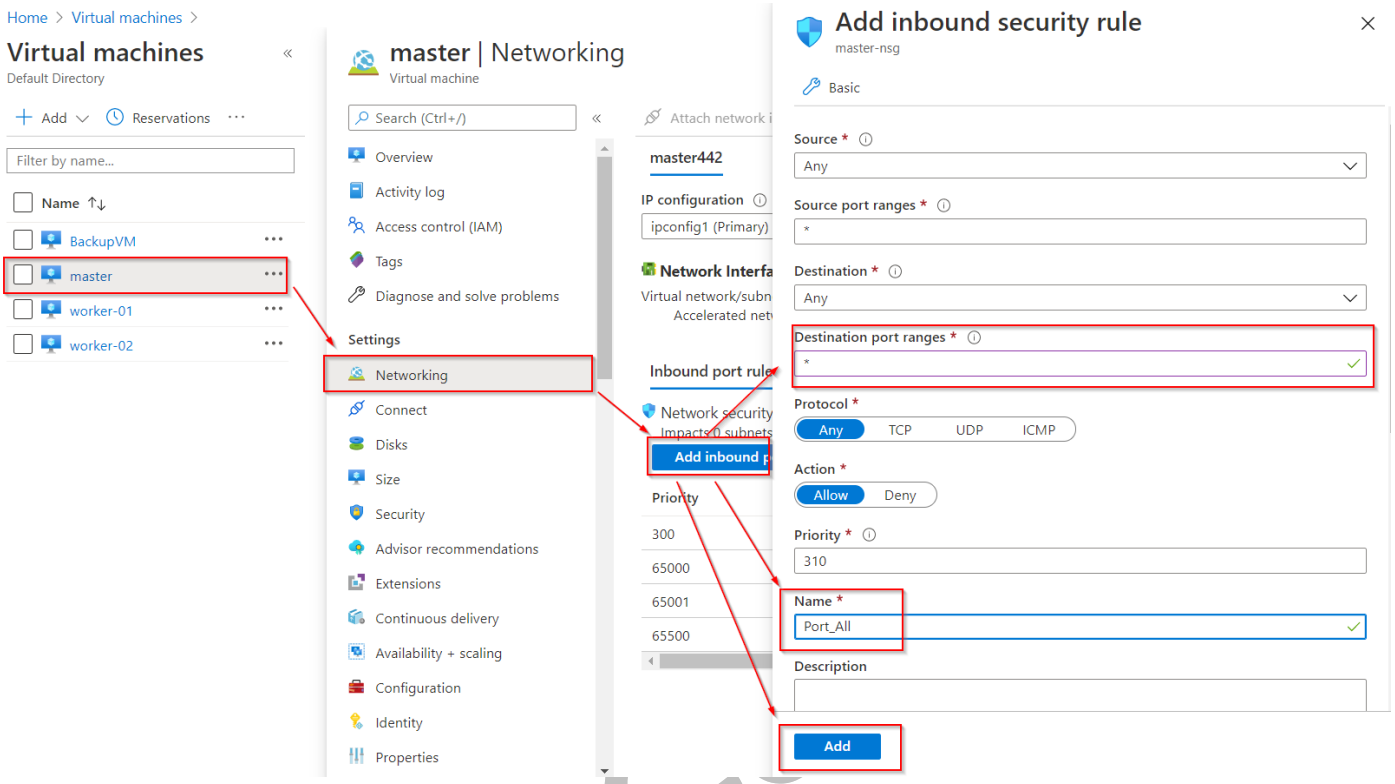
Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

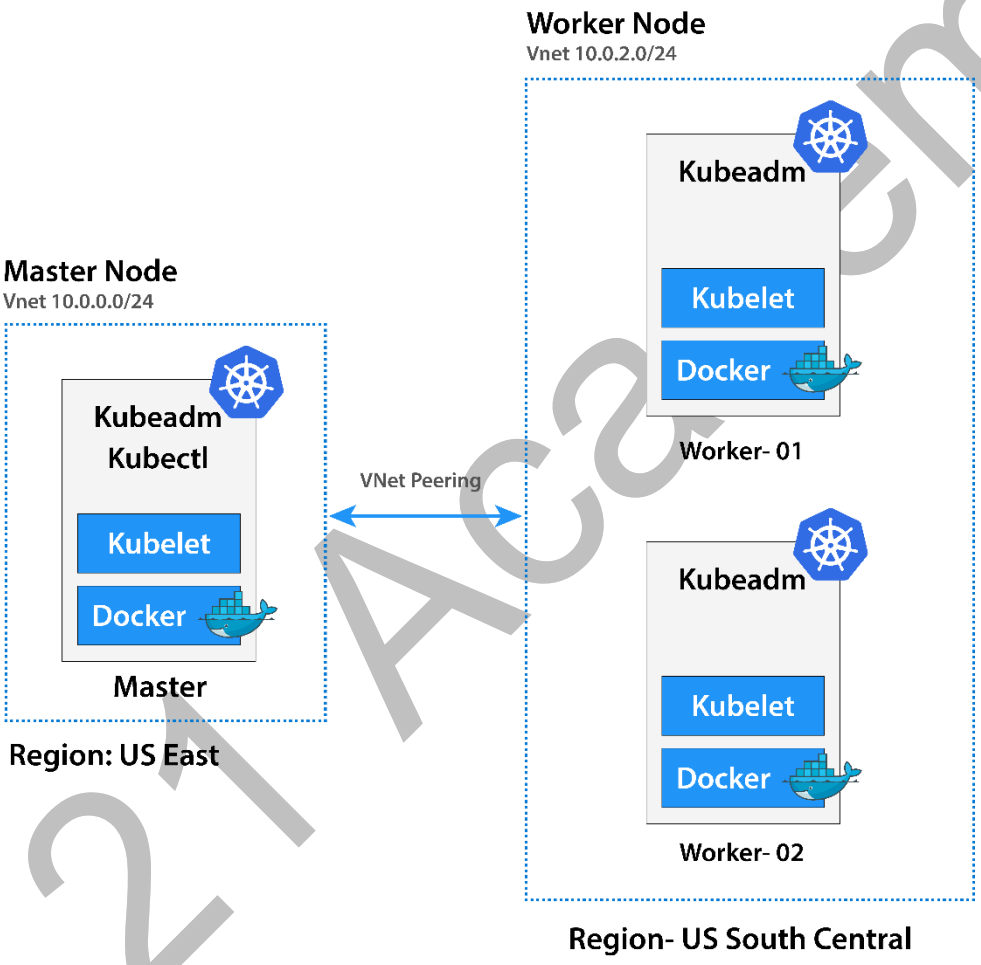
Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

In our labs, we are going to open all ports (as this is play environment) between Master & Worker Node in Azure as given in screenshot below and also covered later in respective section in this guide.

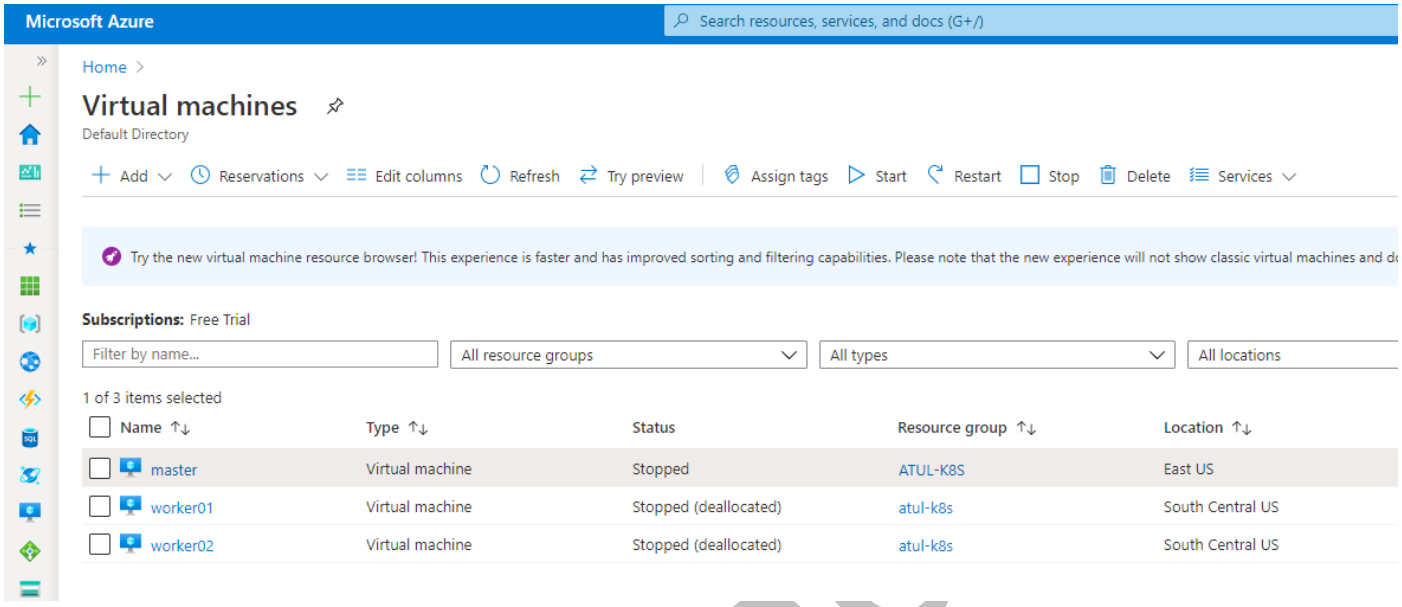

Home > Virtual machines >
Virtual machines
Default Directory
Add
Reservations
Filter by name...
Name
BackupVM
master
worker-01
worker-02
master | Networking
Virtual machine
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Networking
Connect
Disks
Size
Security
Advisor recommendations
Extensions
Continuous delivery
Availability + scaling
Configuration
Identity
Properties
Attach network i
master442
IP configuration
ipconfig1 (Primary)
Network Interfa
Virtual network/subn
Accelerated net
Inbound port rule
Network security
Add inbound p
Priority
300
65000
65001
65500
Add inbound security rule
master-nsg
Basic
Source *
Any
Source port ranges *
*
Destination *
Any
Destination port ranges *
*
Protocol *
Any
TCP
UDP
ICMP
Action *
Allow
Deny
Priority *
310
Name *
Port_All
Description
Add

4.2 Create Master & Worker Node Linux Machines

Ideally to ensure latency between Master & Worer Node is kept to minimum, All cluster members must be created in single Region. In FREE Azure trial account, there is a limit on number of Azure machine you can create hence we create Master Node in one Region and Worker Node in another region and peer these two networks using Azure VNET peering so master & worker nodes can communicate as per screeshot below (Peering is covered later in this guide)



5 CREATE 3 VIRTUAL MACHINES TO SET UP KUBEADM CLUSTER



***Important Note:* In this exercise we are going to create three Virtual Machines. In Azure Free tier account we can't create 3 virtual machine in a Single region due to service limit so we are creating *One Master node in East US Region* and *Two Worker node (worker-1, worker-2 in South central US Region* then we connect *Vnets using VCN Peering* so for now you don't need to convert your account to pay-as you go)**

To Bootstrap a Kubernetes Cluster Using Kubeadm the recommended is to Create 3 new ubuntu VM in Azure Cloud.

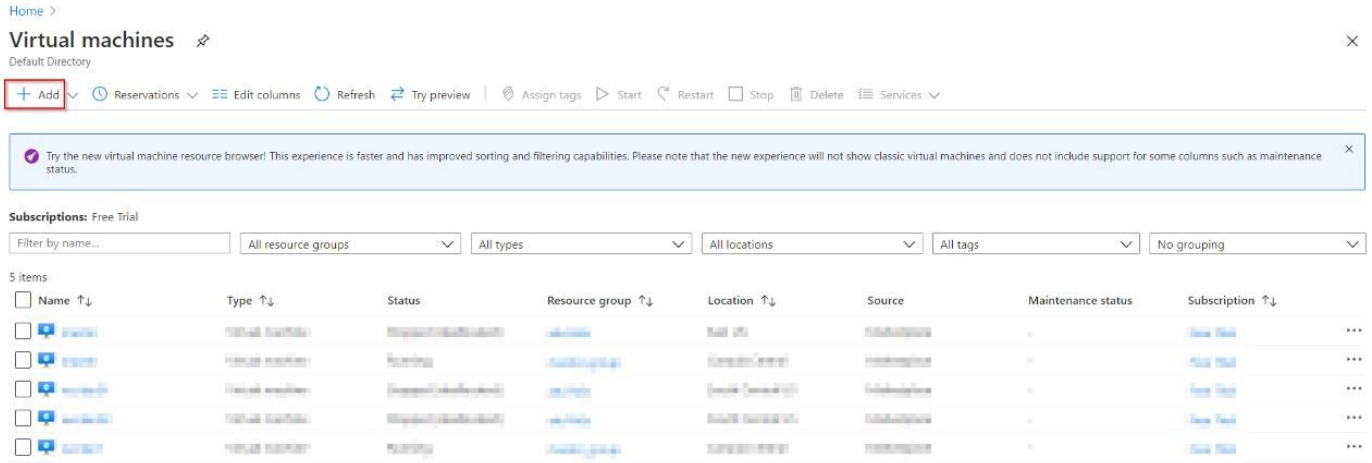
***Note:** One will Work as Master node and other two will work as a Worker Nodes.*

- master
- worker-1
- worker-2

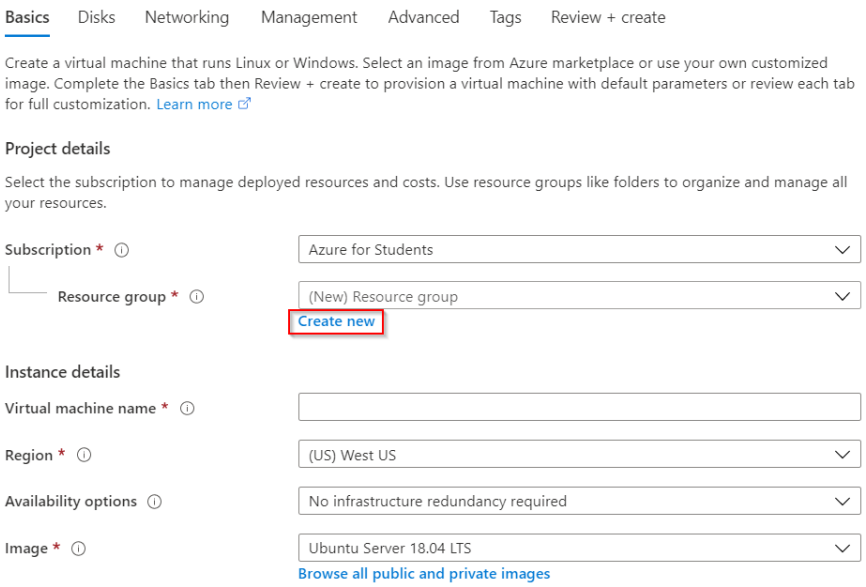
***Note:** Resource Group should be same for all Machines i.e:* **kubeadm**

5.1 Create Master Node In East Region

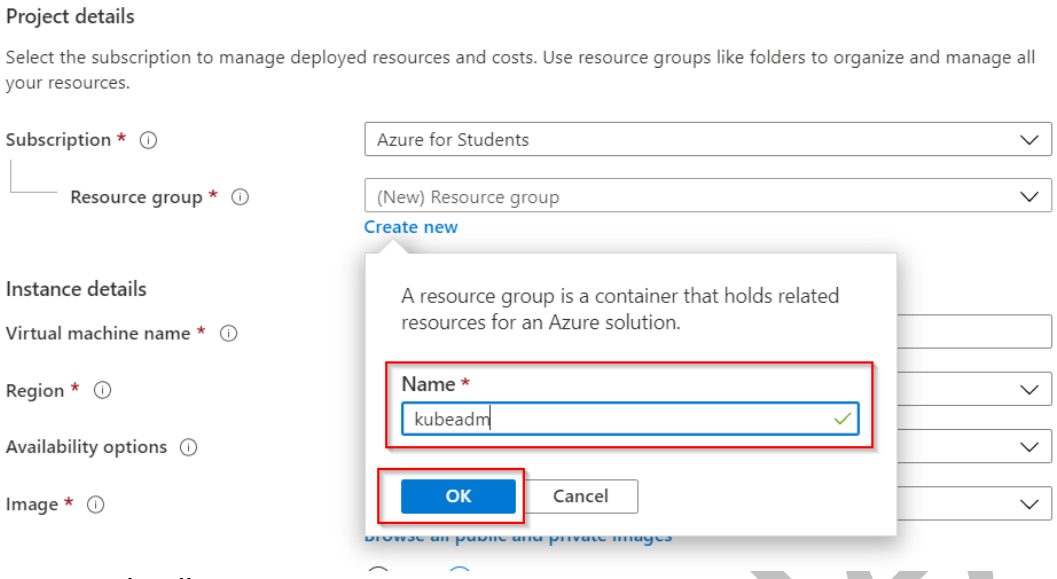
1. Create a Virtual Machine



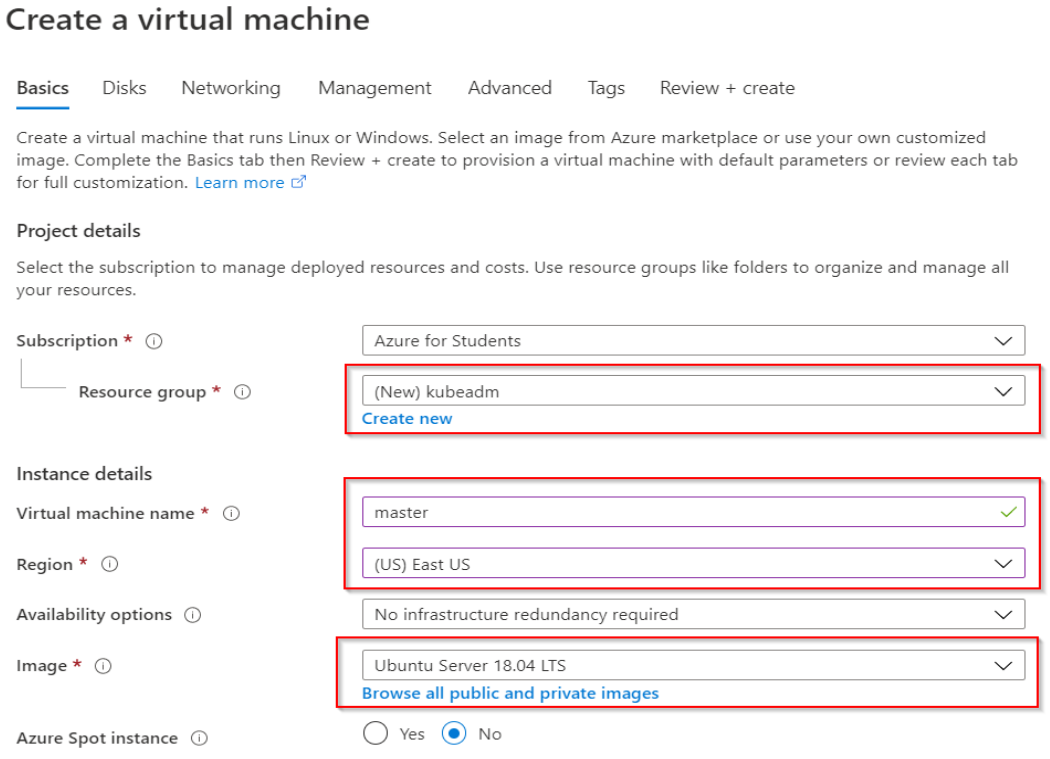
2. Create a new Resource group



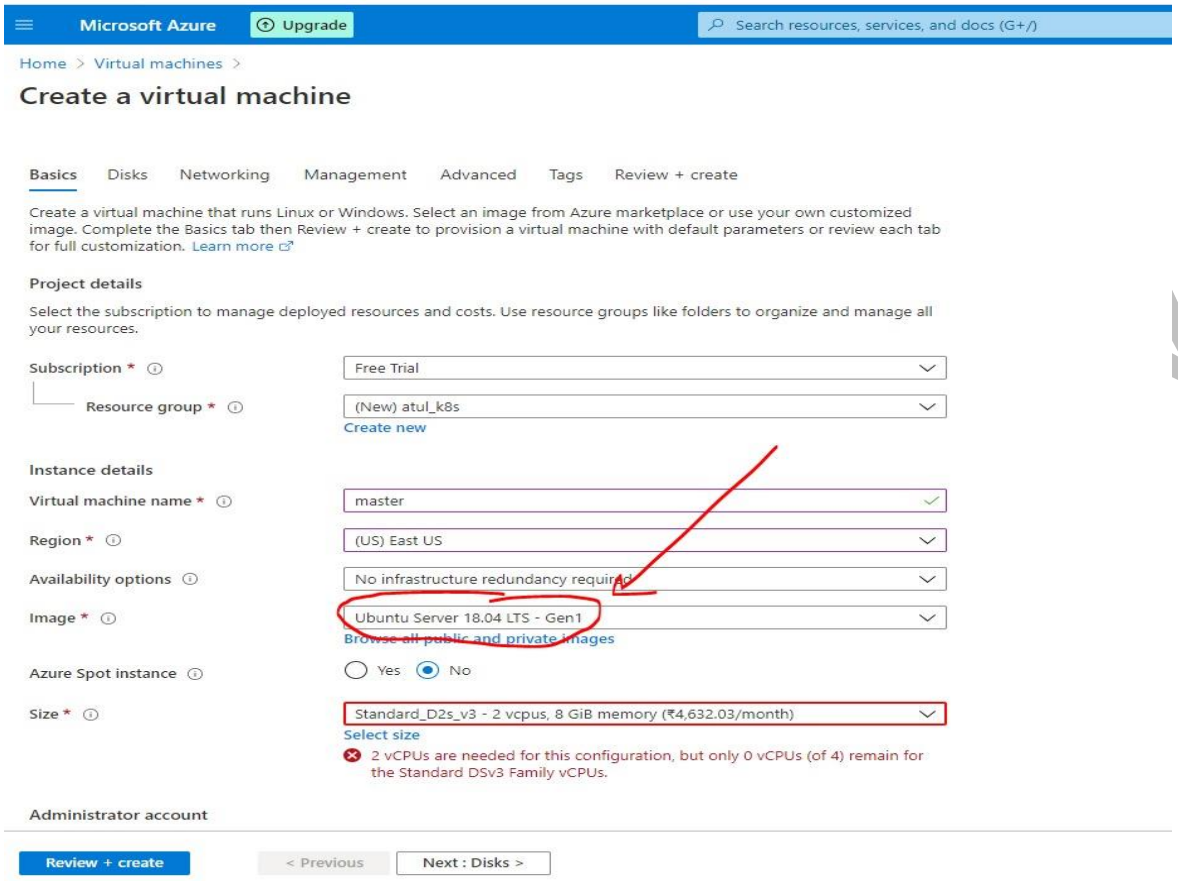
1. **Name:** kubeadm

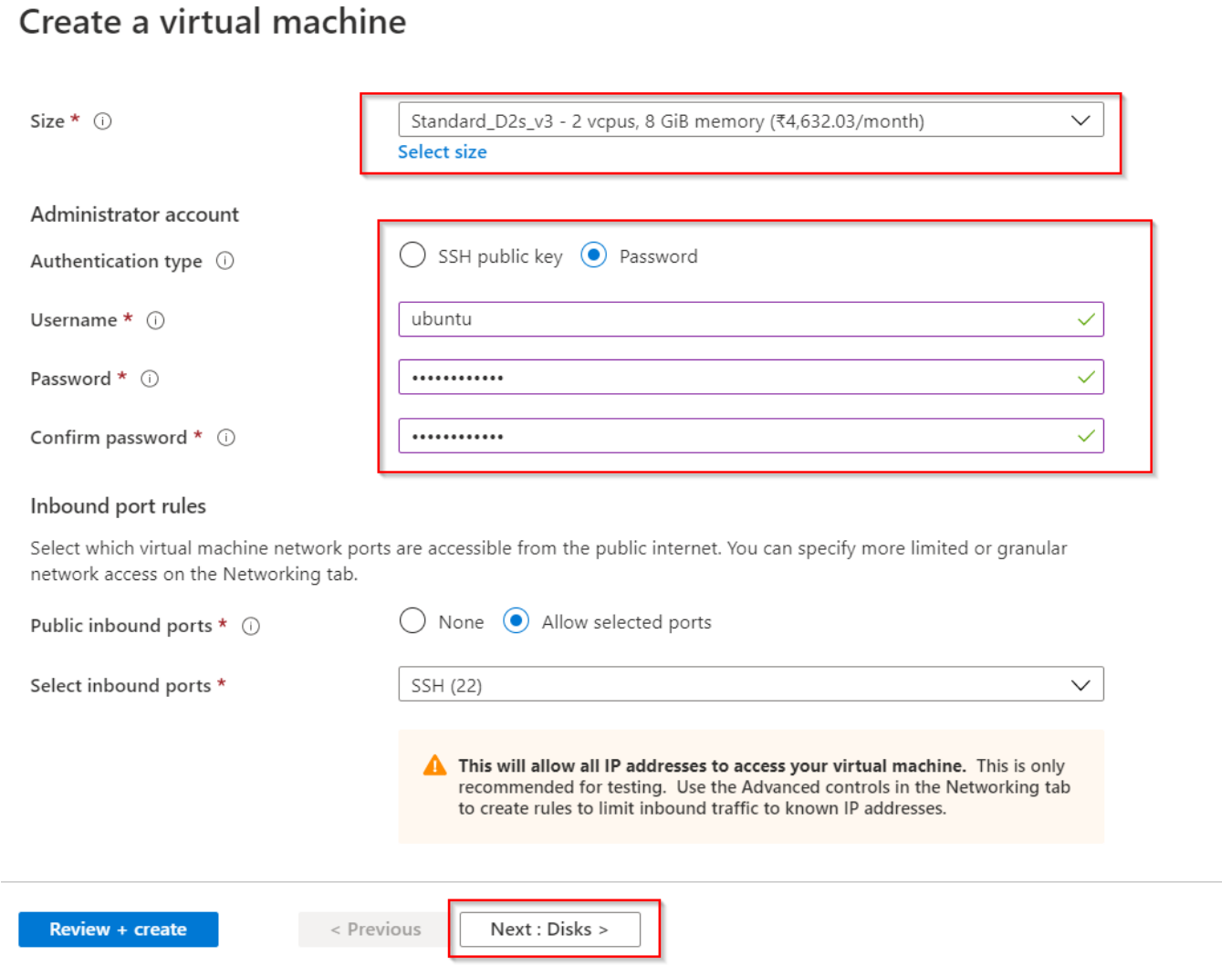


2. Instance details
   - Virtual machine name: **master <Name of your virtual machine>**
   - Region: (US) East US
   - Availability Options: Leave default
   - Image: Leave default



Note: Use latest Image - Ubuntu Server 18.04 LTS – Gen 1


Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > Virtual machines >
Create a virtual machine
Basics
Disks
Networking
Management
Advanced
Tags
Review + create
Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
Free Trial
Resource group *
(New) atul_k8s
Create new
Instance details
Virtual machine name *
master
Region *
(US) East US
Availability options
No infrastructure redundancy required
Image *
Ubuntu Server 18.04 LTS - Gen1
Browse all public and private images
Azure Spot instance
Yes
No
Size *
Standard_D2s_v3 - 2 vcpus, 8 GiB memory (₹4,632.03/month)
Select size
2 vCPUs are needed for this configuration, but only 0 vCPUs (of 4) remain for the Standard DSv3 Family vCPUs.
Administrator account
Review + create
< Previous
Next : Disks >



Same screen go to the top and click/Select **Networking**

Create a virtual machine

Basics | Disks | Networking | Management | Advanced | Tags | Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure for Students

Resource group * : (New) kubeadm

Create new

Instance details

Virtual machine name * : master

Region * : (US) East US

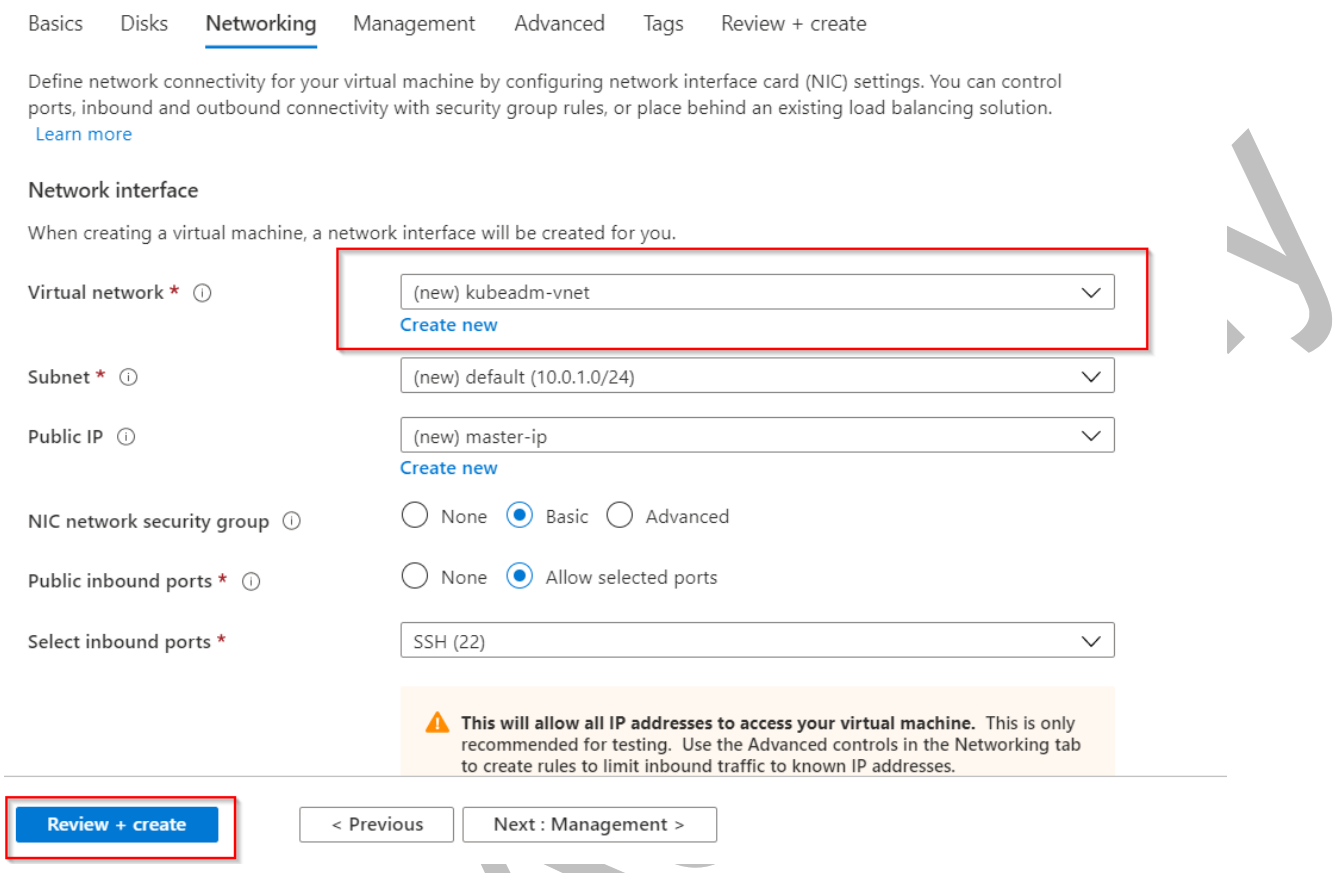
Availability options : No infrastructure redundancy required

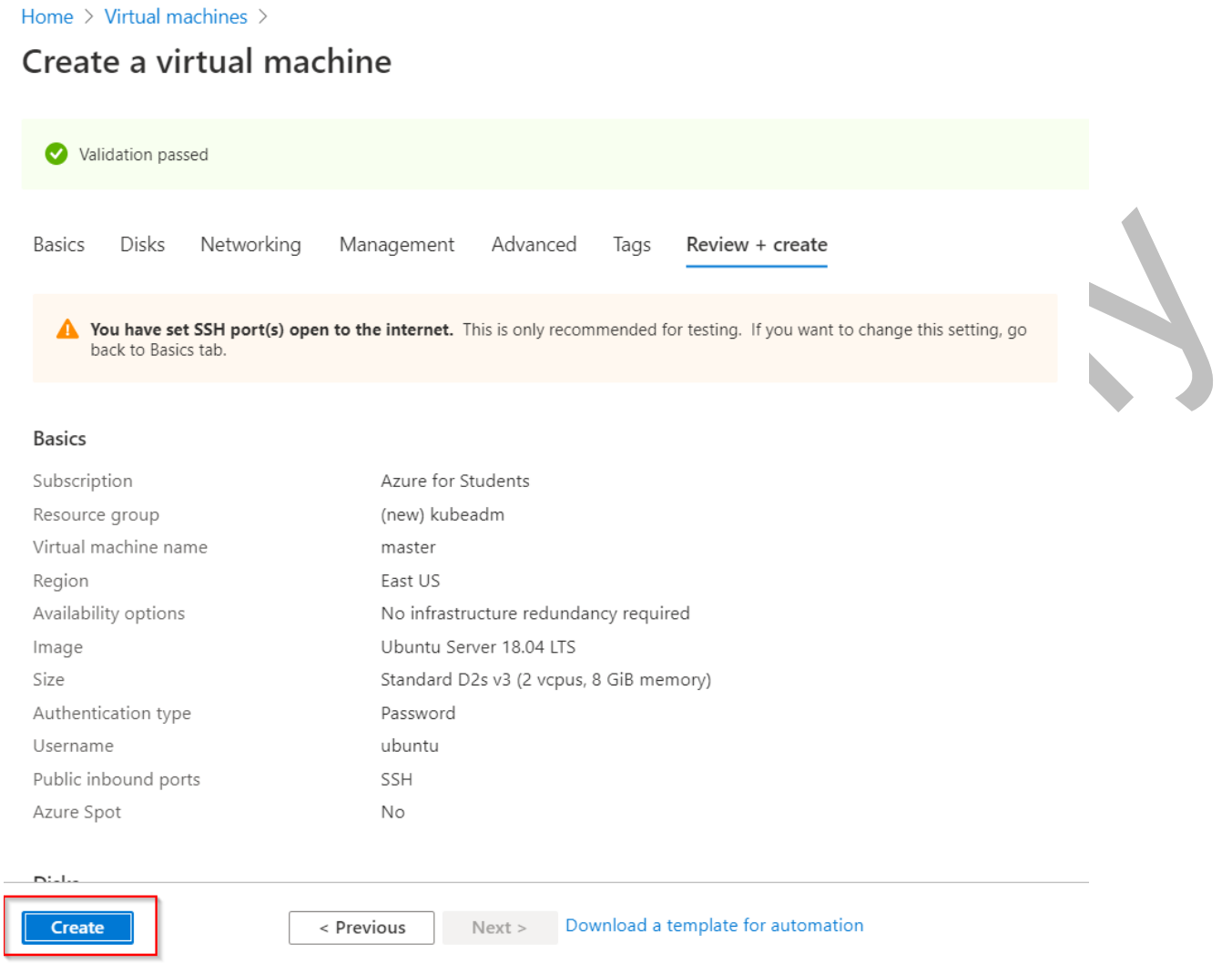
Image * : Ubuntu Server 18.04 LTS

Browse all public and private images

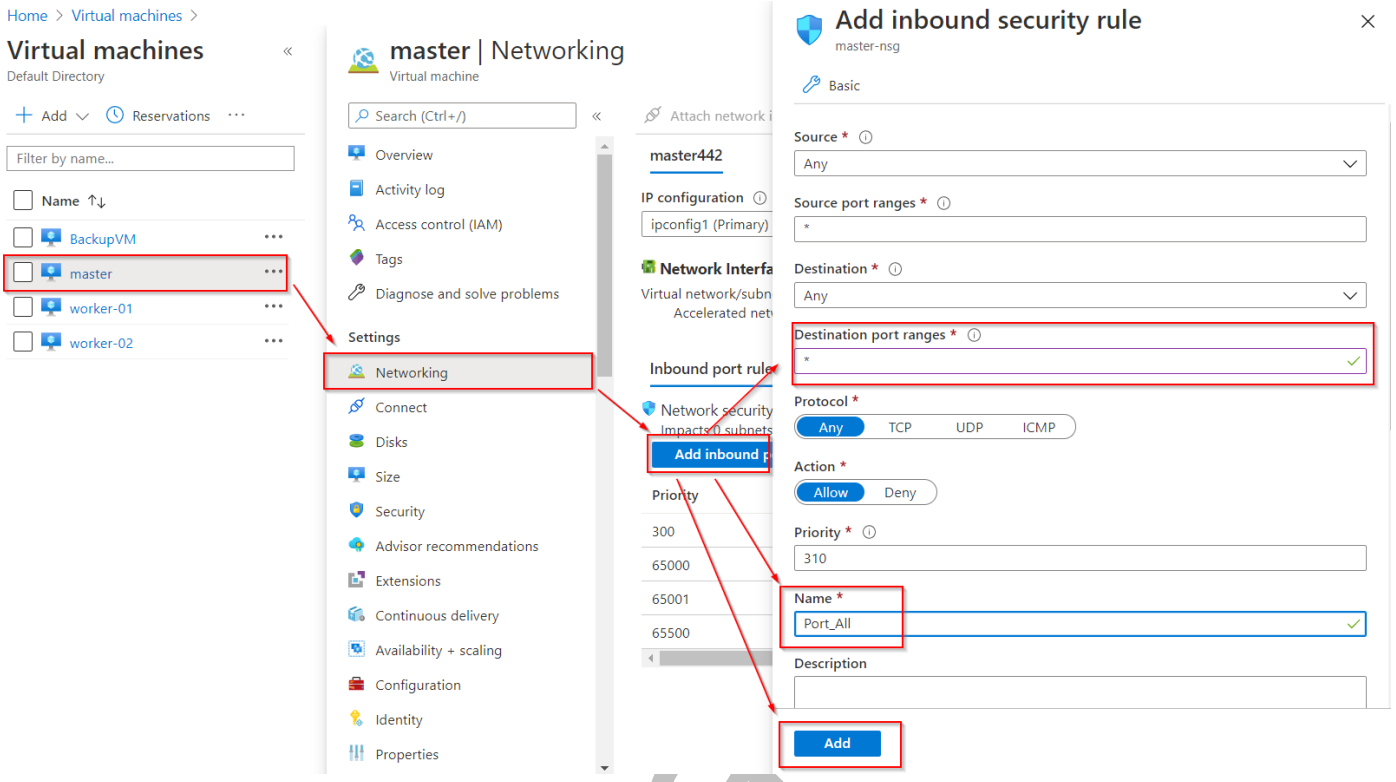
Note: *Note this Virtual Network name we gonna need this in later sections.*

- Click on **Review + Create**



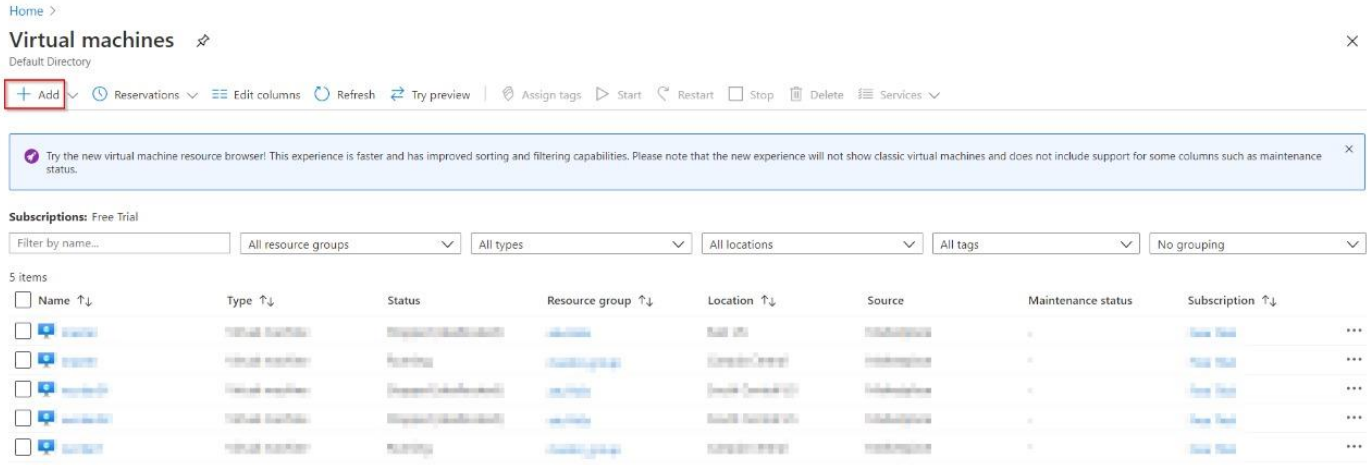


- Open all ports for further labs



5.2 Create Worker Node One In Central US

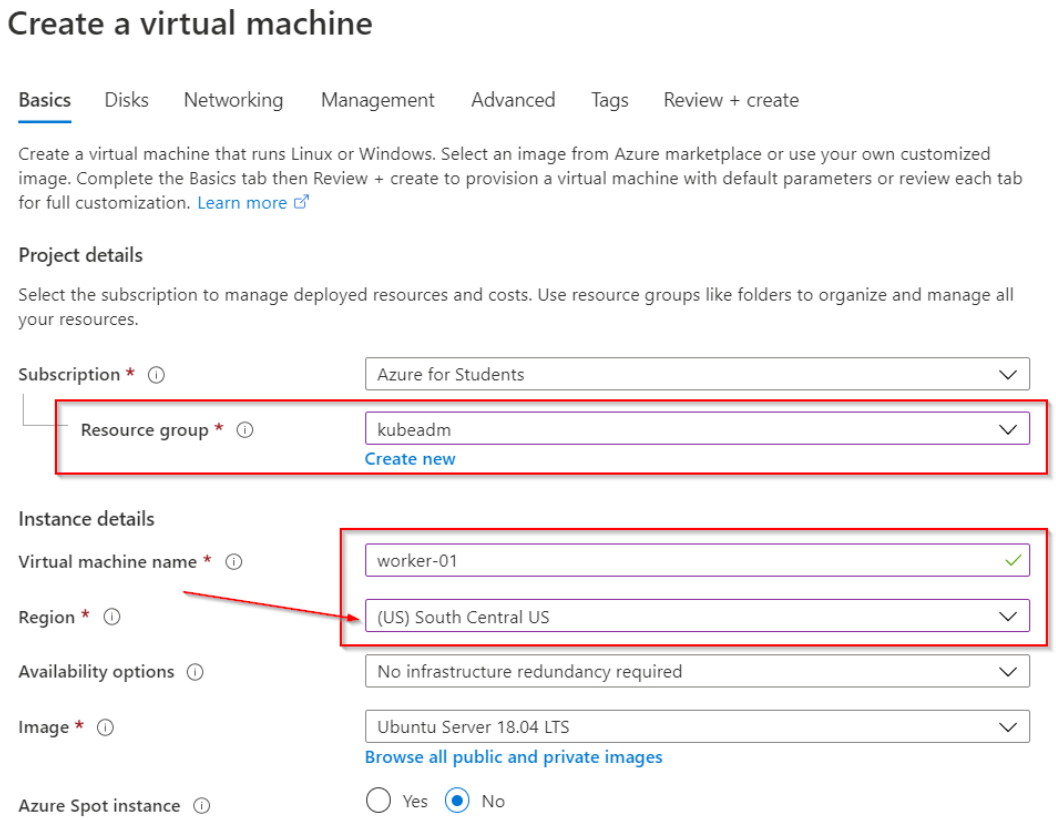
1. Create a Virtual Machine

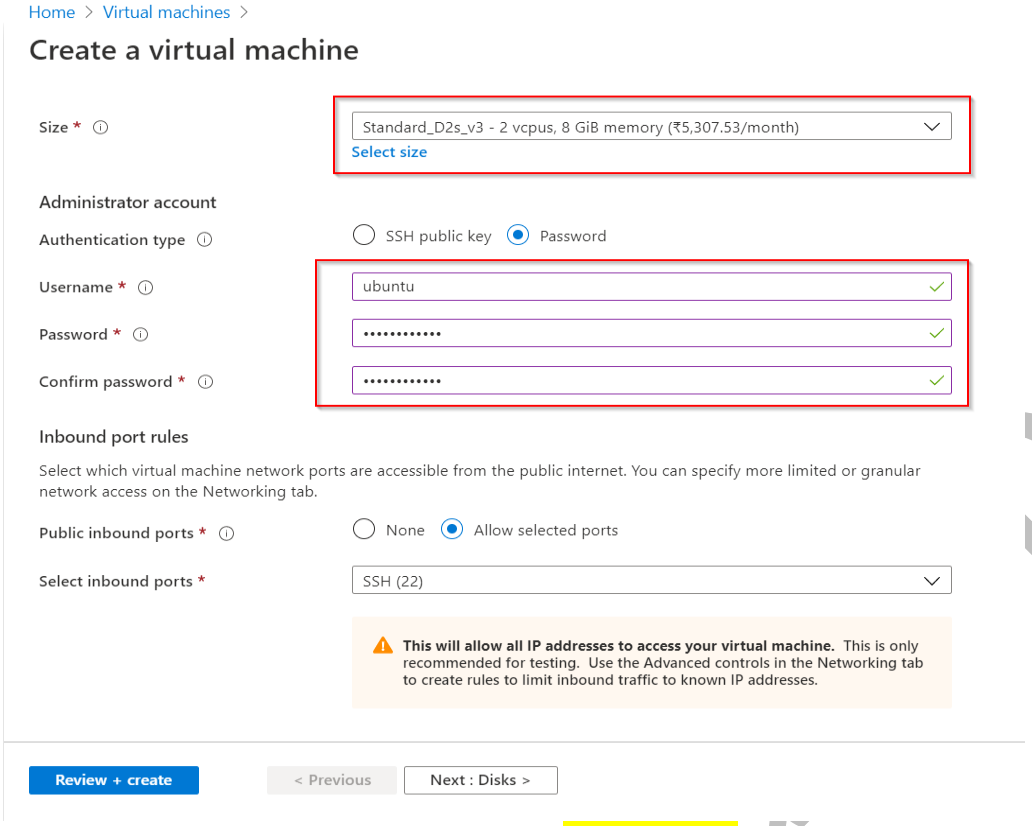


2. Select Resource Group which you created earlier

3. Instance details
   - Virtual machine name: **worker-01 <Name of your virtual machine>**

- Region: (US) South Central US
- Availability Options: Leave default
- Image: Leave default





Same screen go to the top and click/Select **Networking**

Home > Virtual machines >

Create a virtual machine

Basics Disks **Networking** Management Advanced Tags Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.
Learn more

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * : (new) kubeadmvnet485
Create new

Subnet * : (new) default (10.0.2.0/24)

Public IP : (new) worker-01-ip
Create new

NIC network security group : None / Basic / Advanced

Public inbound ports * : None / Allow selected ports

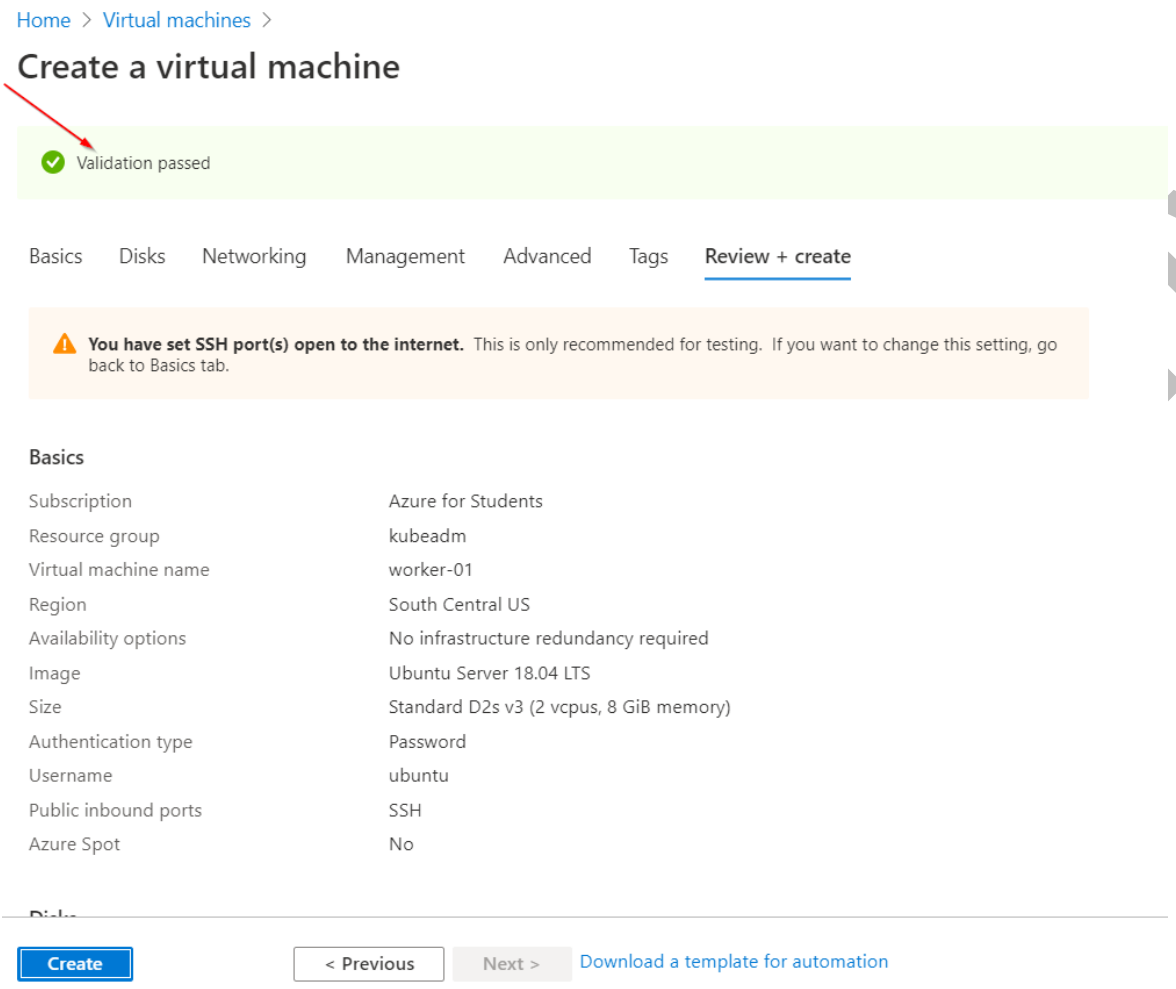
Select inbound ports * : SSH (22)

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

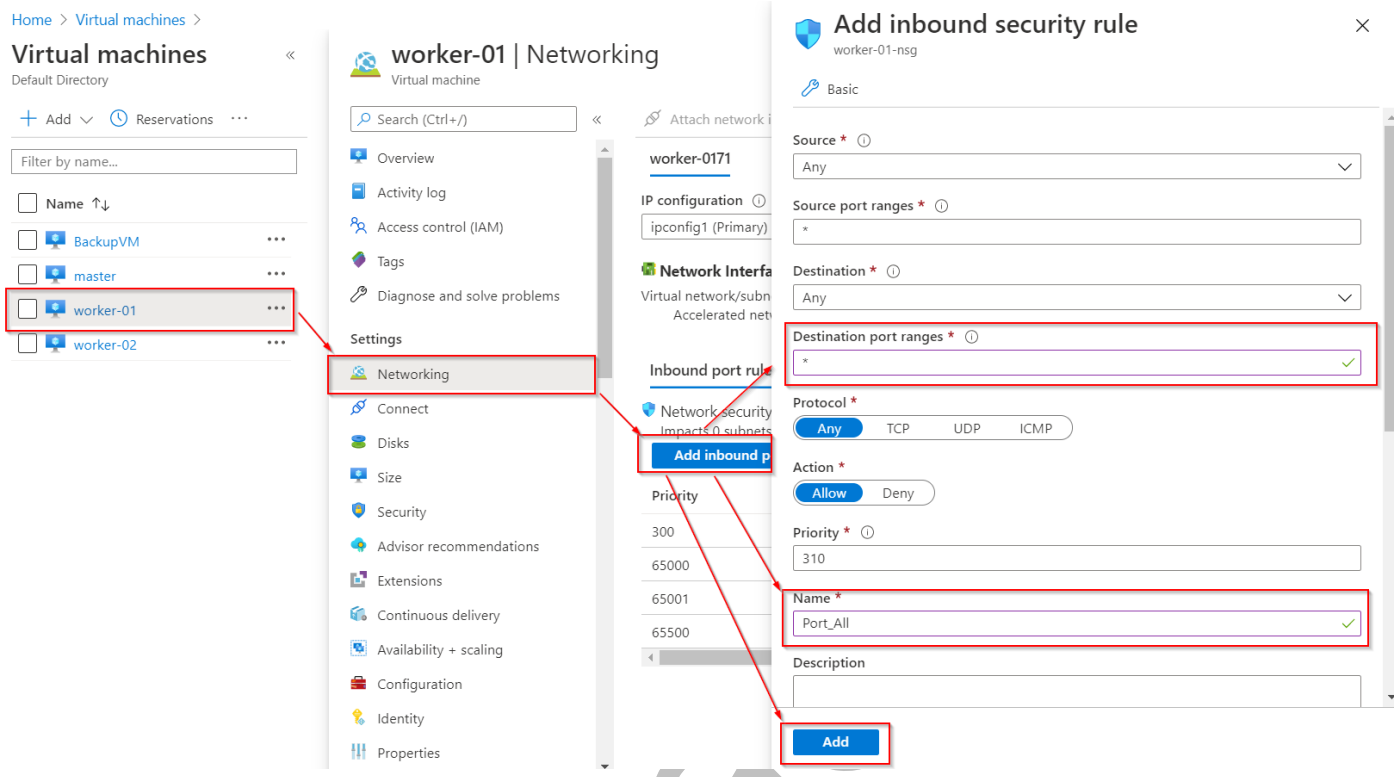
Review + create | < Previous | Next : Management >

Note: *Note this Virtual Network name we gonna need this in later sections.*

- Click on **Review + Create**



- Open all ports fo further labs



5.3 Create Worker Node Two In Central US

Create a virtual machine

Basics | Disks | Networking | Management | Advanced | Tags | Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure for Students

Resource group * : kubeadm

Create new

Instance details

Virtual machine name * : worker-02

Region * : (US) South Central US

Availability options : No infrastructure redundancy required

Image * : Ubuntu Server 18.04 LTS

Browse all public and private images

Azure Spot instance : ○ Yes ◉ No

Home > Virtual machines >

Create a virtual machine

Size * ⓘ — Standard_D2s_v3 - 2 vcpus, 8 GiB memory (₹5,307.53/month)
Select size

Administrator account

Authentication type ⓘ — ○ SSH public key ◉ Password

Username * ⓘ — ubuntu ✓

Password * ⓘ — •••••••••••• ✓

Confirm password * ⓘ — •••••••••••• ✓

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ⓘ — ○ None ◉ Allow selected ports

Select inbound ports * — SSH (22)

⚠ **This will allow all IP addresses to access your virtual machine.** This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Review + create | < Previous | Next : Disks >

Create a virtual machine

Basics | Disks | **Networking** | Management | Advanced | Tags | Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.
Learn more

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * : kubeadmvnet485
Create new

Subnet * : default (10.0.2.0/24)
Manage subnet configuration

Public IP : (new) worker-02-ip
Create new

NIC network security group : None / Basic / Advanced

Public inbound ports * : None / Allow selected ports

Select inbound ports * : SSH (22)

This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Review + create | < Previous | Next : Management >

Home > Virtual machines >

Create a virtual machine

Validation passed

Basics Disks Networking Management Advanced Tags **Review + create**

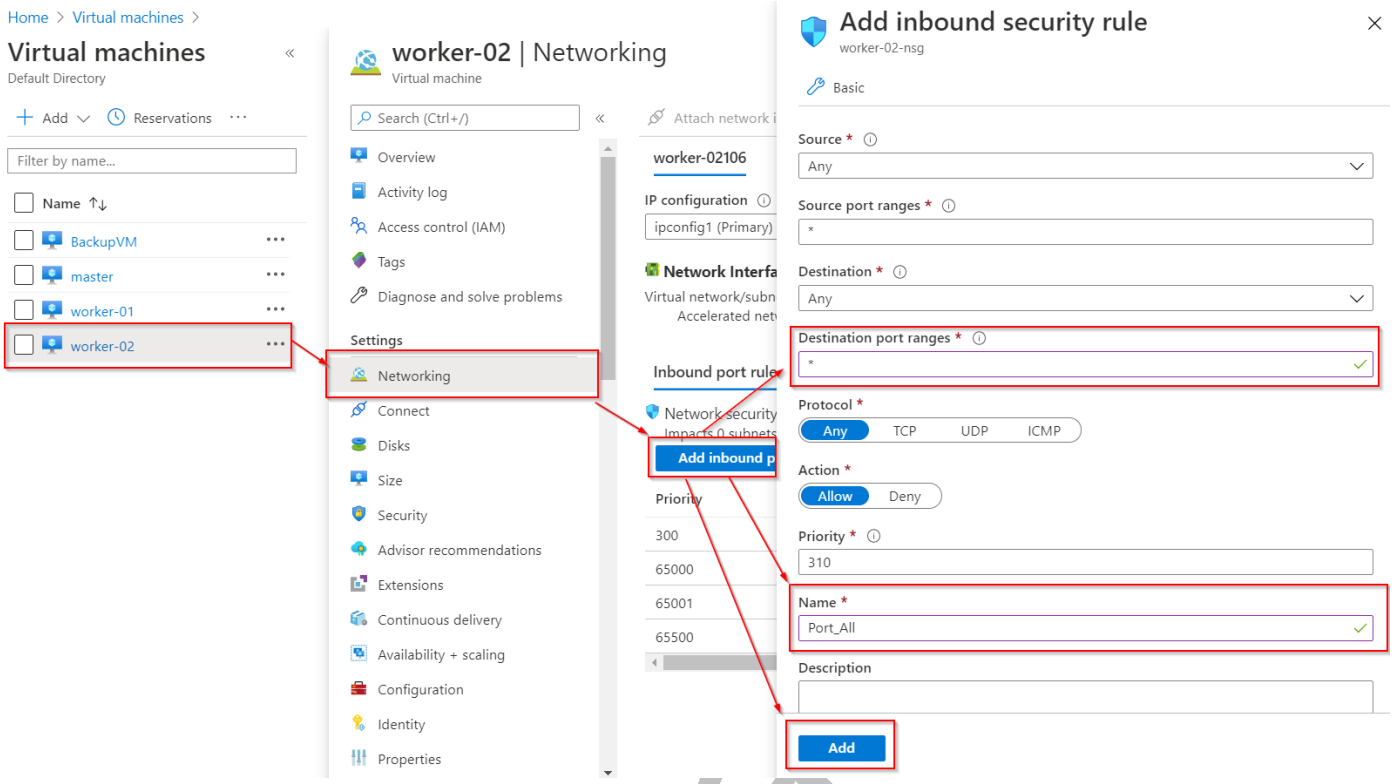
You have set SSH port(s) open to the internet. This is only recommended for testing. If you want to change this setting, go back to Basics tab.

Basics

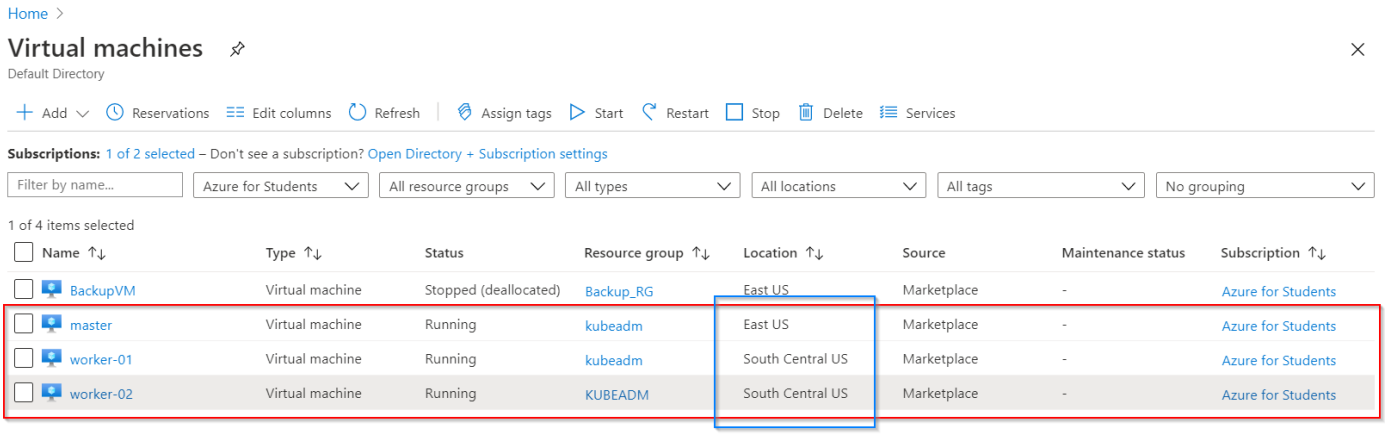
Subscription	Azure for Students
Resource group	kubeadm
Virtual machine name	worker-02
Region	South Central US
Availability options	No infrastructure redundancy required
Image	Ubuntu Server 18.04 LTS
Size	Standard D2s v3 (2 vcpus, 8 GiB memory)
Authentication type	Password
Username	ubuntu
Public inbound ports	SSH
Azure Spot	No

Create < Previous Next > Download a template for automation

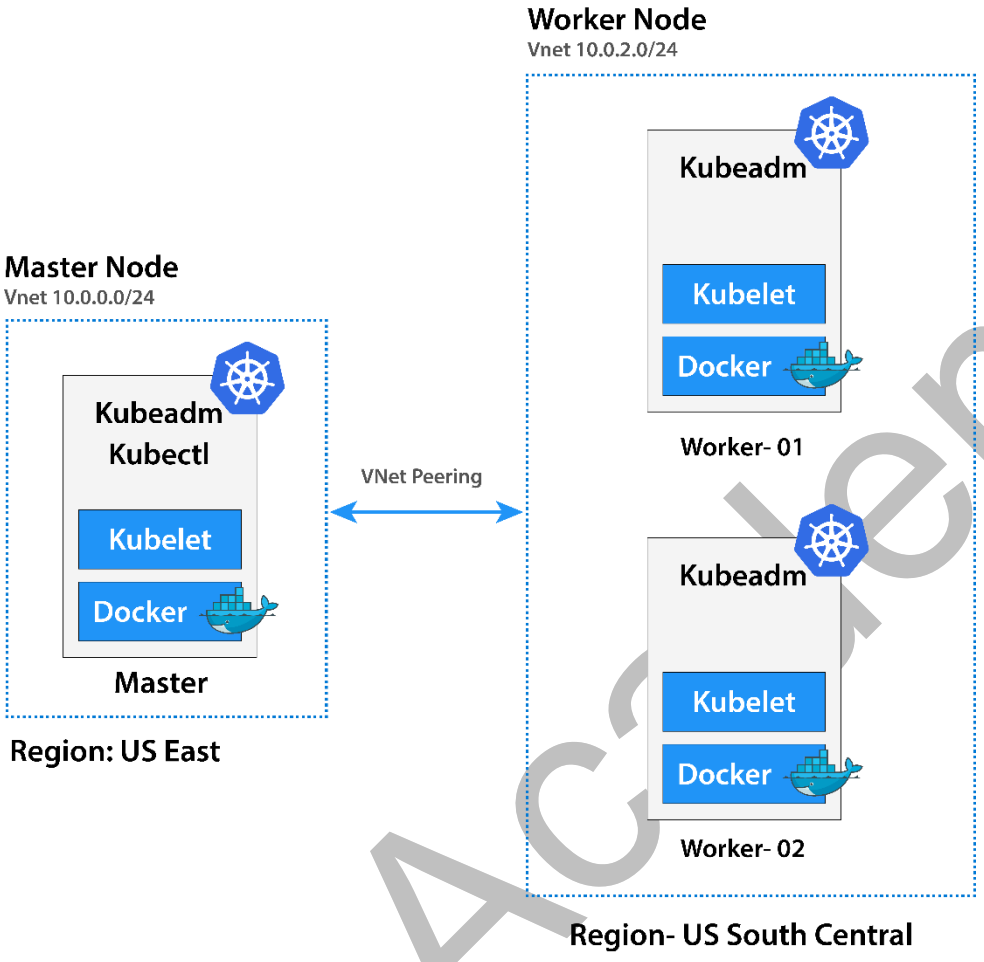
- Open all ports for further labs



- Final Screen after Creating all the Machine

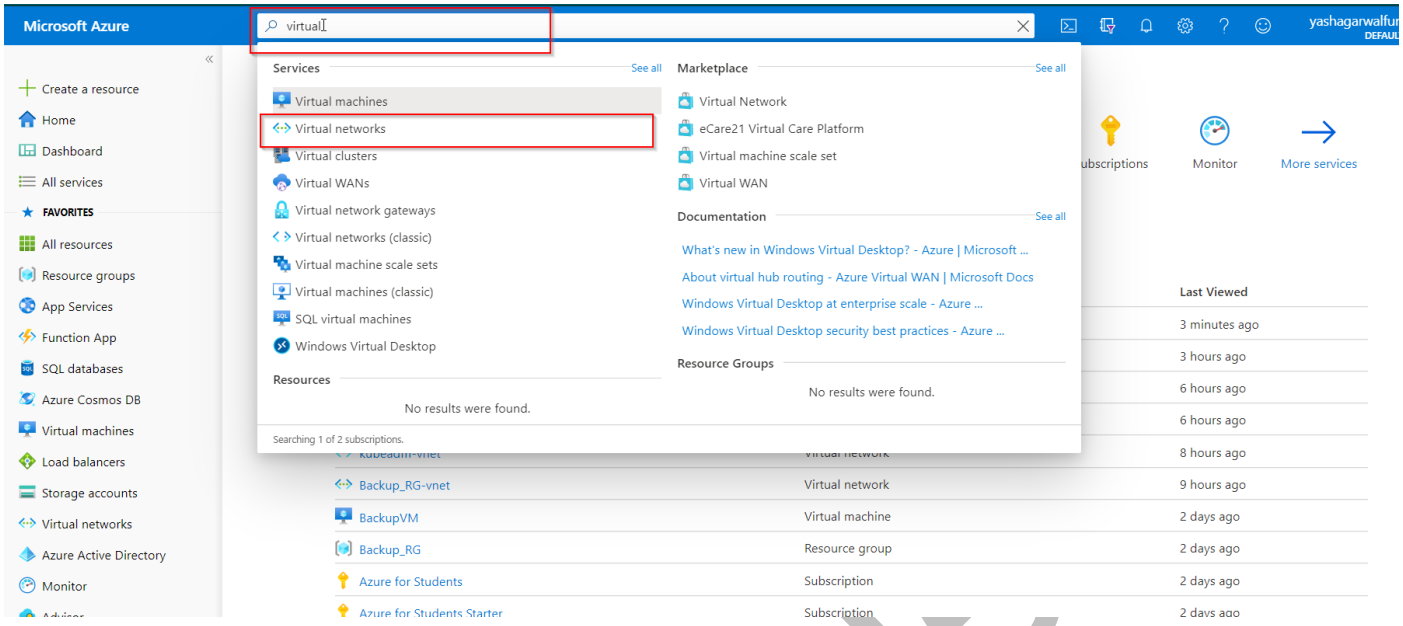


5.4 Virtual Network Peering

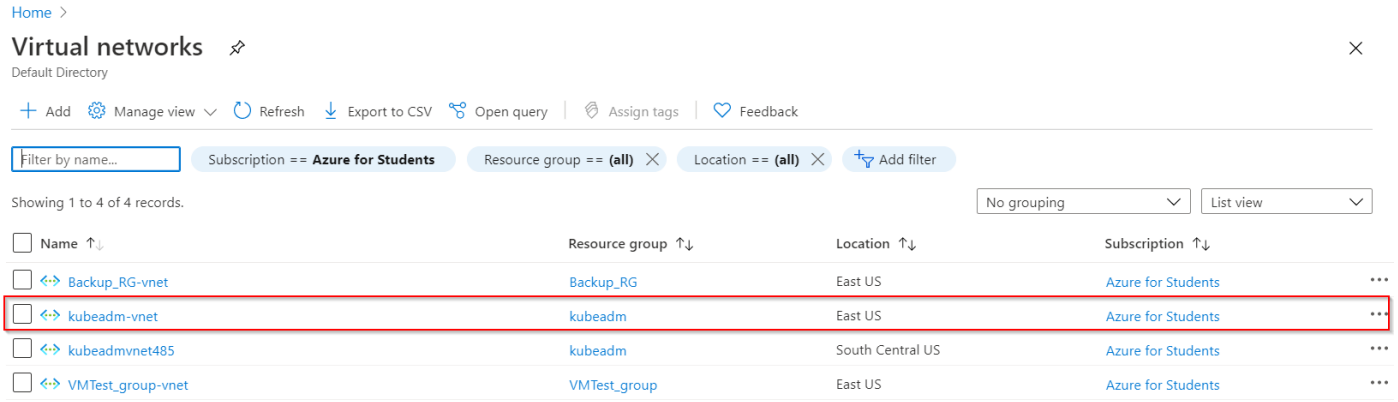


Important Note: *Now We will perform virtual peering so the machines in different regions can talk to each other.*

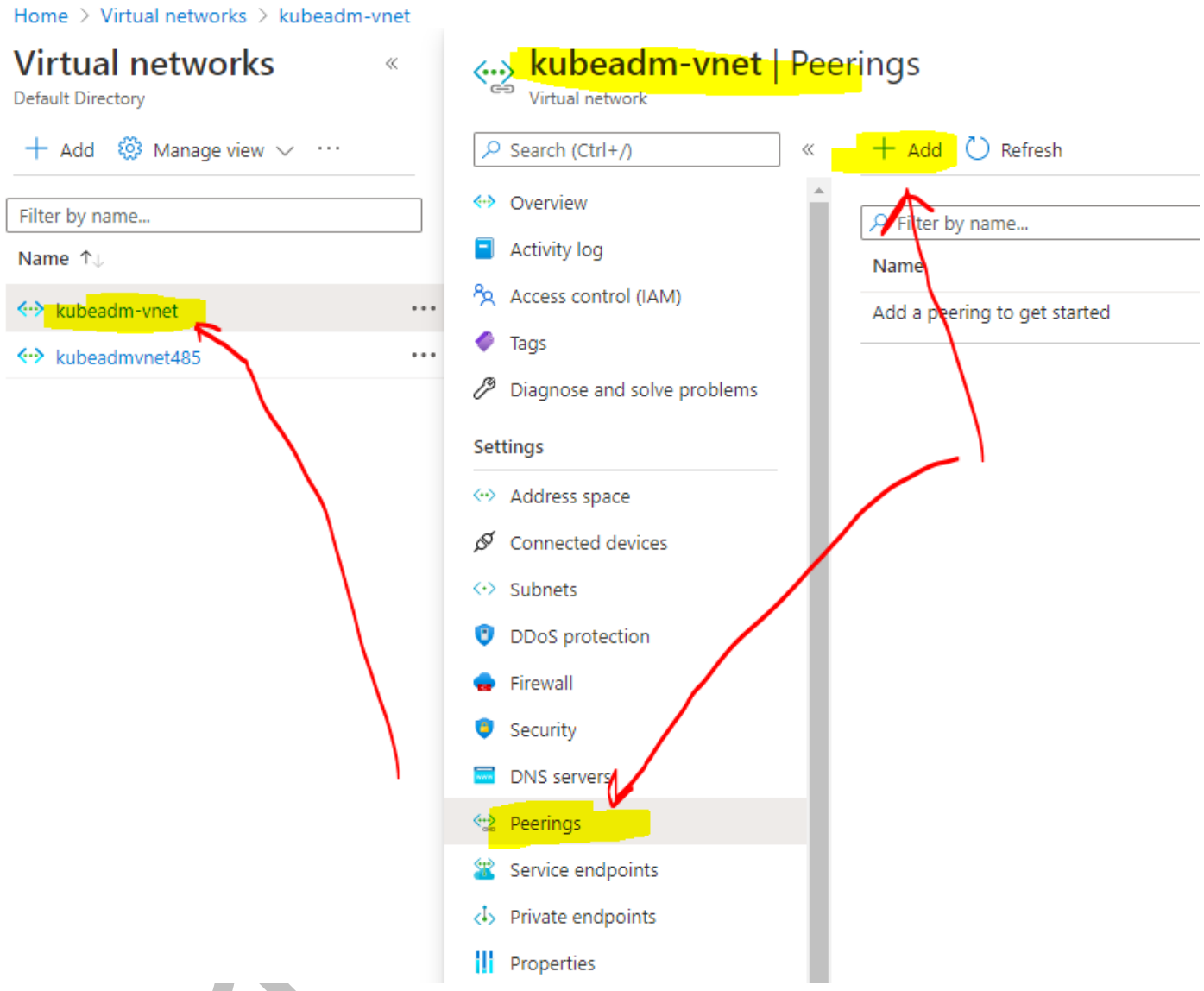
1. In search bar search for virtual network



2. Select master virtual machine Virtual Network



3. In Overview section go for **Peering** and click **Add**

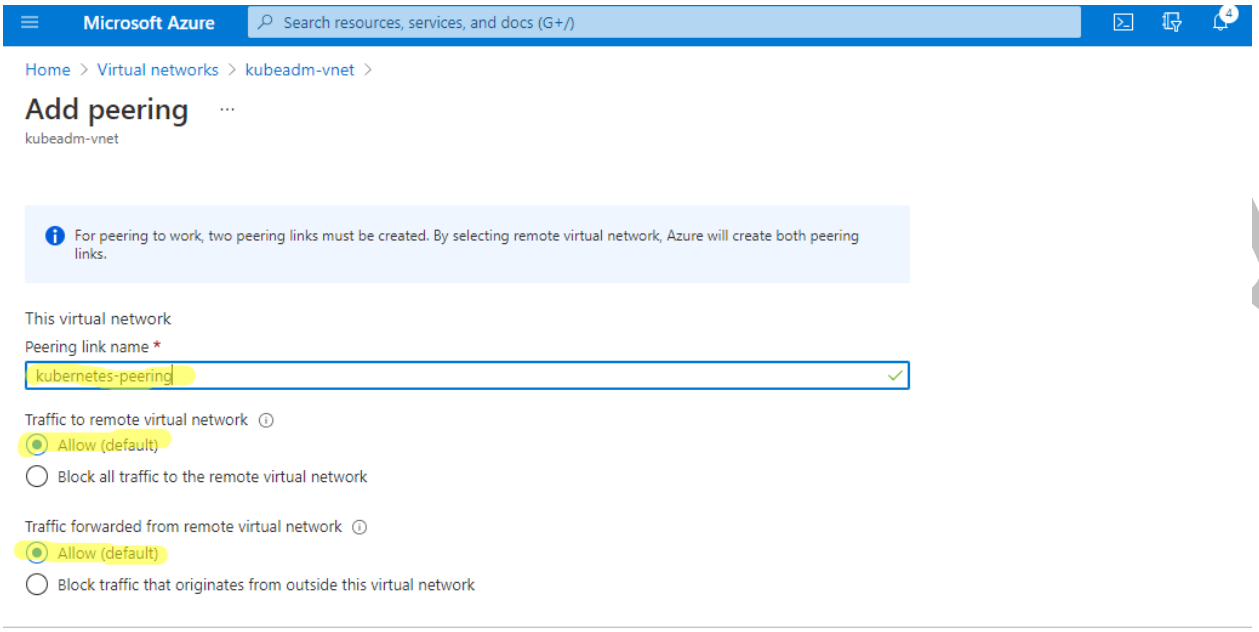


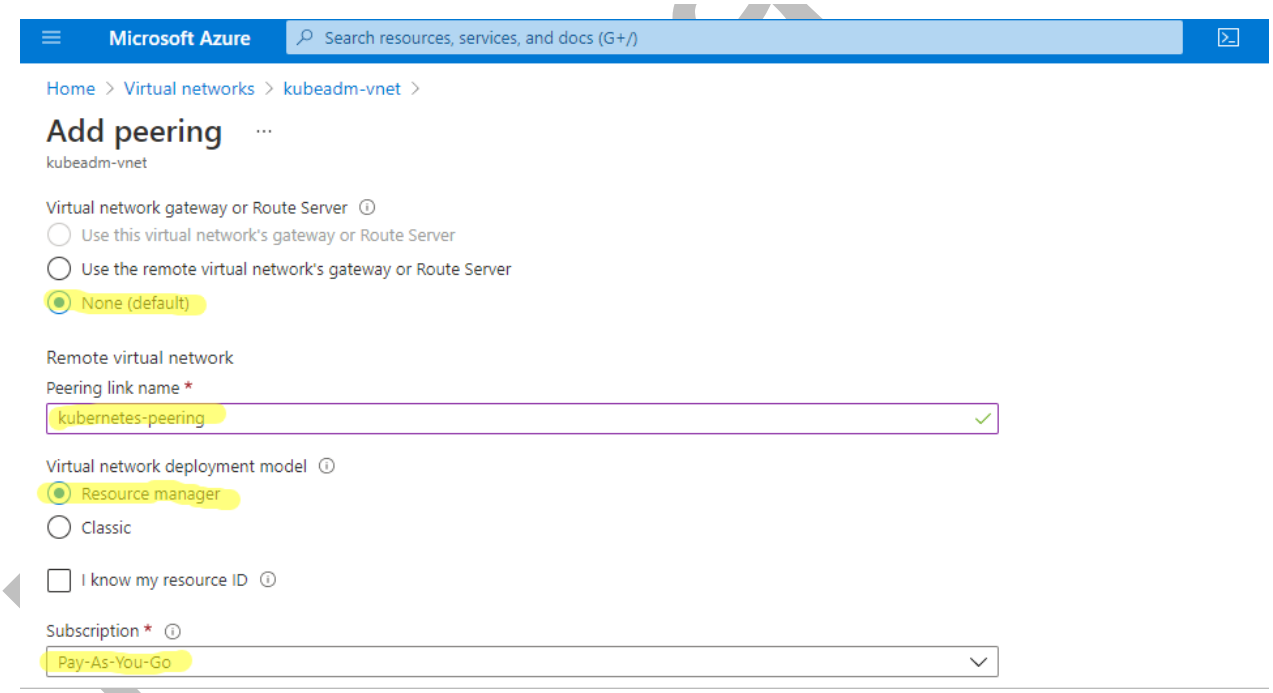
4. On **Add Peering** screen Fill:

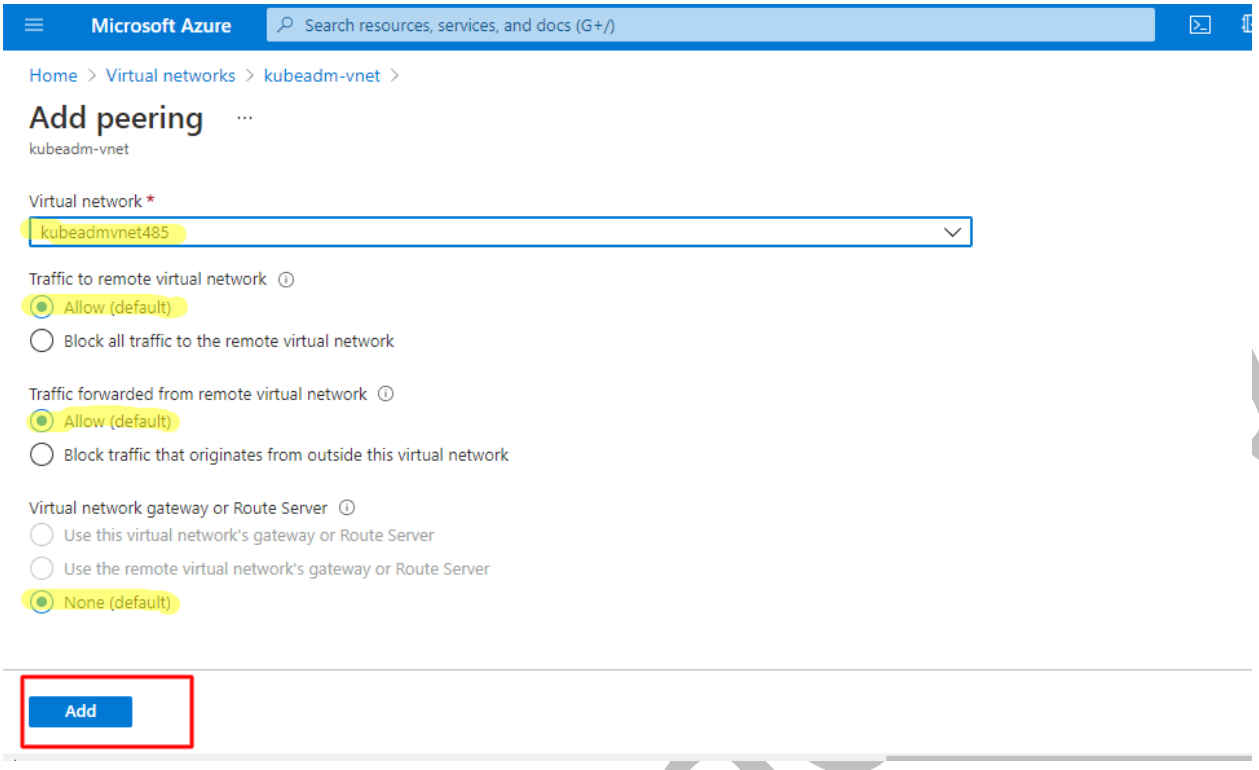
- **Peering link name:** kubertnetes-peering
- **Traffic to remote virtual network:** choose Allow (default)
- **Traffic forwarded from remote virtual network:** choose Allow (default)
- **Virtual network gateway**: Choose None (default)

- **Virtual Network: Kubeadmvnet485** (*This is the other network that you want to peer, in our case network where our Worker node VMs are created*)

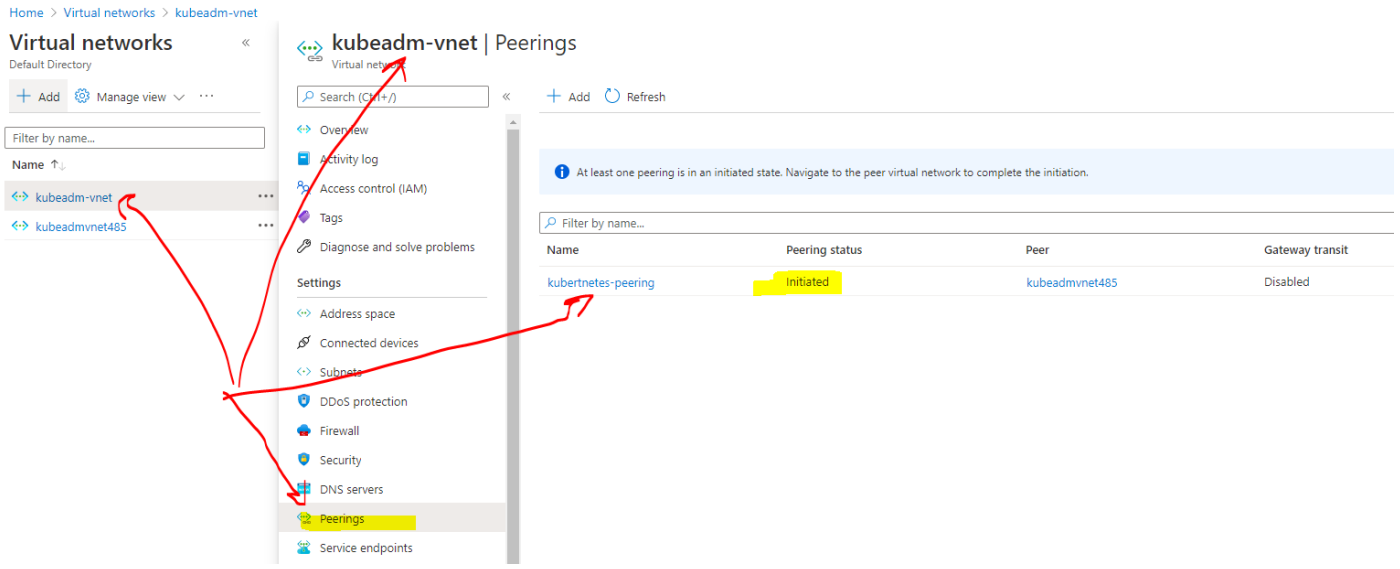
Note: **Select same options as given in the Image.**



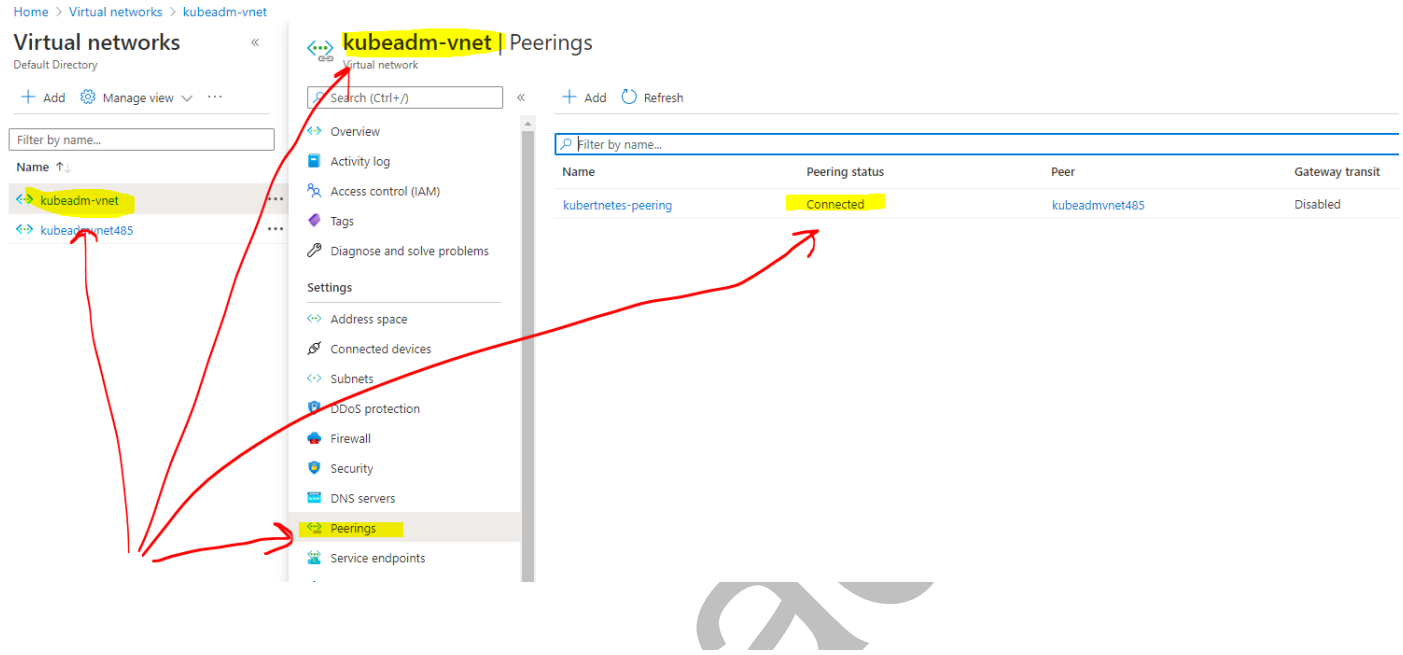


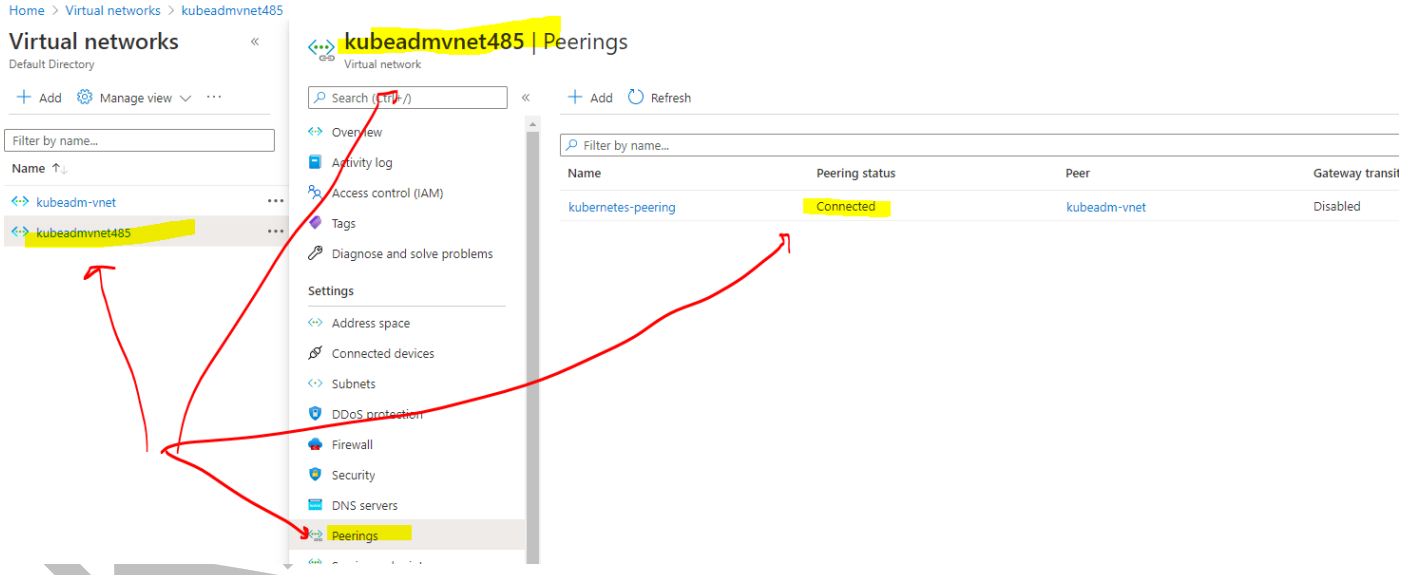


5. Click on add, you will see Peering status as **Initiated**



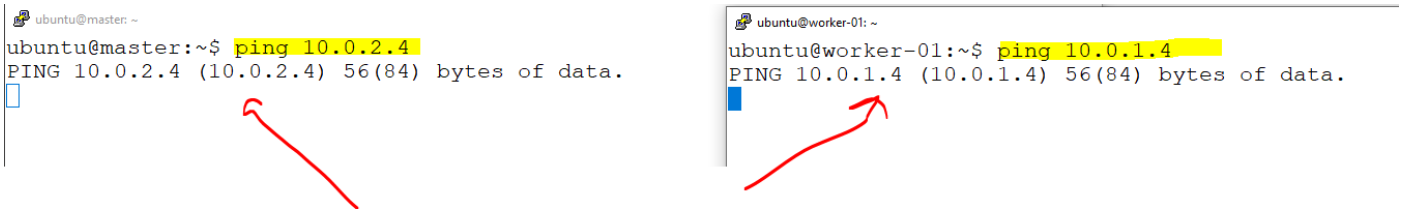
6. Ensure that status of Peering is Connected





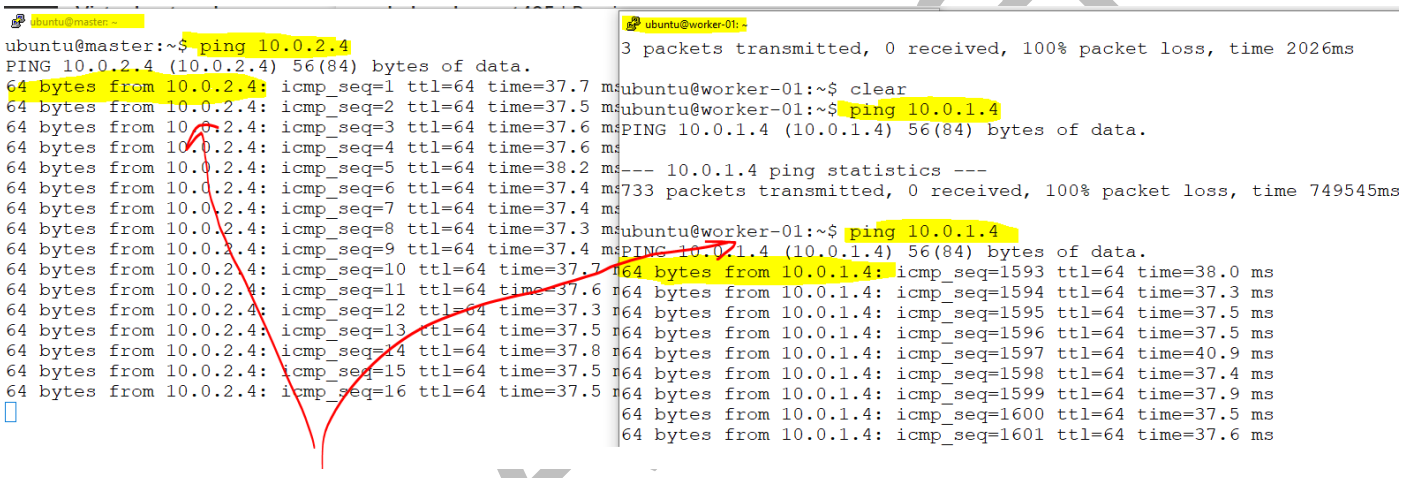
7. Test connectivity by pinging machine's IP in different network

a) Without peering these machines won't be able to ping each other over their private Ips



b) After peering enabled machines in different Network will be able to ping each other over their private IPs

Note: Here 10.0.2.4 is Private IP of worker node that is accessible from master node and 10.0.1.4 is Private IP of Master node that is accessible from worker node



This completes VNet peering

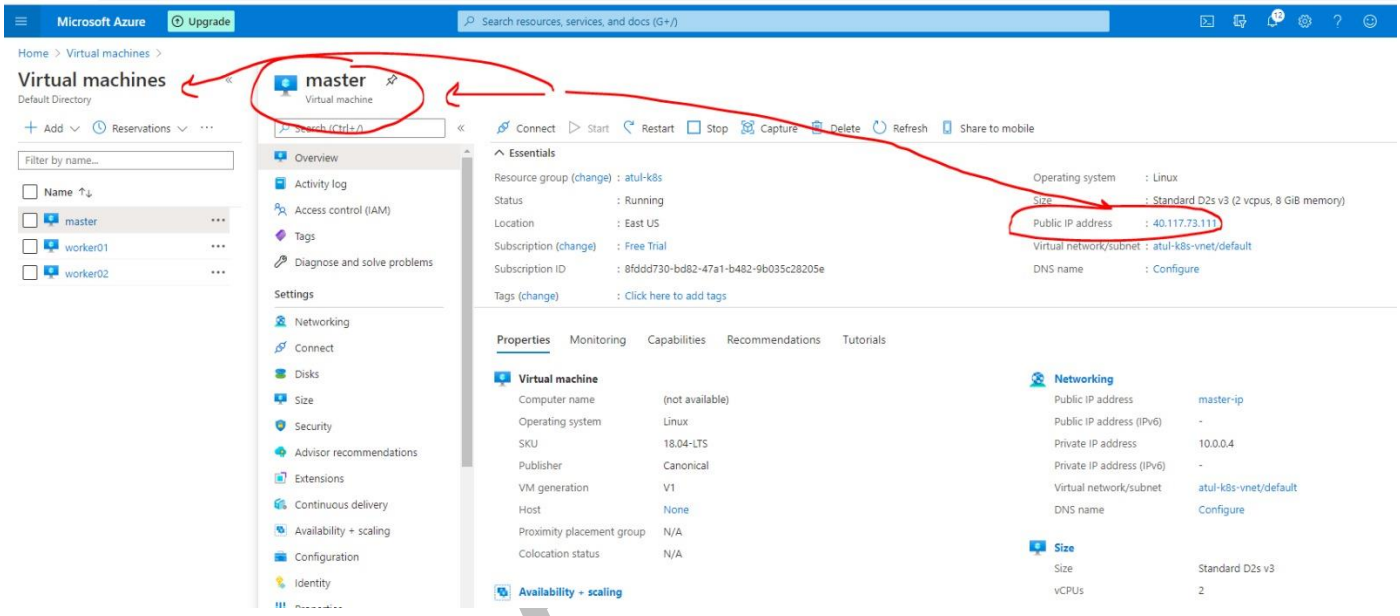
6 INSTALLING DOCKER, KUBEADM AND OTHER KUBECTL PACKAGES

***Note:** First perform All Section 6 Steps on the **Master node** then repet same on the **both worker Nodes too**.*

1. SSH to the virtual machine with the username and password you used while creating the VM

```
$ ssh root@publicipaddress
```

Note: To get the Public IP Address, go to the master node and copy the Public IP Address.



Note: Linux or Mac user can use ssh command. Windows user can use Putty.

2. Switch to root user in case you aren't logged as root

```
$ sudo -i
```

```
ubuntu@kubeadm-master:~$
ubuntu@kubeadm-master:~$ sudo su
root@kubeadm-master:/home/ubuntu#
```

3. Install docker package using the following command

```
$ apt-get update && apt-get install -y docker.io
```

```
root@kubeadm-master:/home/ubuntu# apt-get update && apt-get install -y docker.io
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
```

Verify the docker version installed

```
$ docker --version
```

4. Update the apt package index and install packages needed to use the Kubernetes apt repository:

```
$ apt-get update && apt-get install -y apt-transport-https ca-certificates curl
```

```
root@Master:~# sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [48.5 kB]
Fetched 221 kB in 0s (459 kB/s)
Reading package lists... Done
root@Master:~# sudo apt-get install -y apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20210119~18.04.1).
ca-certificates set to manually installed.
apt-transport-https is already the newest version (1.6.14).
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-151
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libcurl4
The following packages will be upgraded:
  curl libcurl4
2 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
Need to get 378 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 curl amd64 7.58.0-2ubuntu3.14 [159 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcurl4 amd64 7.58.0-2ubuntu3.14 [219 kB]
Fetched 378 kB in 0s (16.8 MB/s)
(Reading database ... 77278 files and directories currently installed.)
Preparing to unpack .../curl_7.58.0-2ubuntu3.14_amd64.deb ...
Unpacking curl (7.58.0-2ubuntu3.14) over (7.58.0-2ubuntu3.13) ...
Preparing to unpack .../libcurl4_7.58.0-2ubuntu3.14_amd64.deb ...
Unpacking libcurl4:amd64 (7.58.0-2ubuntu3.14) over (7.58.0-2ubuntu3.13) ...
Setting up libcurl4:amd64 (7.58.0-2ubuntu3.14) ...
Setting up curl (7.58.0-2ubuntu3.14) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
```

5. Download the Google Cloud public signing key:

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

6. Add the Kubernetes apt repository:

Note: *Copy-Paste or type Next Command Carefully*

```
$ 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
```

```
root@Master:~# echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/so
urces.list.d/kubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main
```

7. Update apt package index, install kubelet, kubeadm and kubectl, and pin their version:

```
$ KUBE_VERSION=1.21.0
$ apt-get update
```

```
root@Master:~# KUBE_VERSION=1.21.0
root@Master:~# apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Err:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY FEEA9169307EA071 NO_PUBKEY 8B57C5C2836F4BEB
Reading package lists... Done
W: GPG error: https://packages.cloud.google.com/apt kubernetes-xenial InRelease: The following signatures couldn't be verified because the public key is not avai
lable: NO_PUBKEY FEEA9169307EA071 NO_PUBKEY 8B57C5C2836F4BEB
E: The repository 'https://apt.kubernetes.io kubernetes-xenial InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
root@Master:~#
```

```
$ apt-get install -y kubelet=${KUBE_VERSION}-00  kubeadm=${KUBE_VERSION}-00 kubectl=${KUBE_VERSION}-00 kubernetes-cni=0.8.7-00
```

```
root@Master:~# apt-get install -y kubelet=${KUBE_VERSION}-00  kubeadm=${KUBE_VERSION}-00 kubectl=${KUBE_VERSION}-00 kubernetes-cni=0.8.7-00
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-151
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  conntrack cri-tools socat
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 7 newly installed, 0 to remove and 13 not upgraded.
Need to get 70.4 MB of archives.
After this operation, 309 MB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 conntrack amd64 1:1.4.4+snapshot20161117-6ubuntu2 [30.6 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 socat amd64 1.7.3.2-2ubuntu2 [342 kB]
Get:3 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-tools amd64 1.13.0-01 [8775 kB]
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubernetes-cni amd64 0.8.7-00 [25.0 MB]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.21.0-00 [18.8 MB]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.21.0-00 [8972 kB]
Get:7 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.21.0-00 [8544 kB]
Fetched 70.4 MB in 2s (30.1 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 77278 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.4+snapshot20161117-6ubuntu2_amd64.deb ...
Unpacking conntrack (1:1.4.4+snapshot20161117-6ubuntu2) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.13.0-01_amd64.deb ...
Unpacking cri-tools (1.13.0-01) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../2-kubernetes-cni_0.8.7-00_amd64.deb ...
```

```
$ apt-mark hold kubelet kubeadm kubectl
```

8. Configure cgroup driver and storage driver.

```
$ cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "storage-driver": "overlay2"
}
EOF
$ mkdir -p /etc/systemd/system/docker.service.d
```

```
root@worker01:~# cat > /etc/docker/daemon.json <<EOF
> {
>   "exec-opts": ["native.cgroupdriver=systemd"],
>   "log-driver": "json-file",
>   "storage-driver": "overlay2"
> }
> EOF
root@worker01:~# mkdir -p /etc/systemd/system/docker.service.d
root@worker01:~#
```

9. Restart the docker

```
$ systemctl daemon-reload
$ systemctl restart docker && systemctl enable docker
```

10. Verify Storage driver and cgroup driver

```
$ docker info | grep -i "storage"
$ docker info | grep -i "cgroup"
```

```
root@Master:~# docker info | grep -i "storage"
 Storage Driver: overlay2
WARNING: No swap limit support
root@Master:~# docker info | grep -i "cgroup"
WARNING: No swap limit support
 Cgroup Driver: systemd
 Cgroup Version: 1
```

11. Start the kubelet service is required on all the nodes

```
$ systemctl enable kubelet && systemctl start kubelet
```

7 CONFIGURE FIREWALL, UPDATE IPTABLES SETTINGS, DISABLE SELINUX AND DISABLE SWAP

NOTE: This section is optional. In Ubuntu all this steps are preconfigured so need to perform in ubuntu OS. But if you are using other linux distribution.You May require to use the following steps.

7.1 Configure Firewall

To fulfil their tasks, nodes, containers, and pods must be able to communicate across the cluster. Add the following ports by entering the listed commands.

On the Master Node enter

```
$ sudo firewall-cmd --permanent --add-port=6443/tcp
$ sudo firewall-cmd --permanent --add-port=2379-2380/tcp
$ sudo firewall-cmd --permanent --add-port=10250/tcp
$ sudo firewall-cmd --permanent --add-port=10251/tcp
$ sudo firewall-cmd --permanent --add-port=10252/tcp
$ sudo firewall-cmd --permanent --add-port=10255/tcp
$ sudo firewall-cmd –reload
```

Each time a port is added the system confirms with a ‘success’ message.

```
Master@Master:~$ sudo firewall-cmd --permanent --add-port=6443/tcp
Warning: ALREADY_ENABLED: 6443:tcp
success
Master@Master:~$ sudo firewall-cmd --permanent --add-port=2379-2380/tcp
Warning: ALREADY_ENABLED: 2379-2380:tcp
success
Master@Master:~$ sudo firewall-cmd --permanent --add-port=10250/tcp
Warning: ALREADY_ENABLED: 10250:tcp
success
Master@Master:~$ sudo firewall-cmd --permanent --add-port=10251/tcp
Warning: ALREADY_ENABLED: 10251:tcp
success
Master@Master:~$ sudo firewall-cmd --permanent --add-port=10252/tcp
Warning: ALREADY_ENABLED: 10252:tcp
success
Master@Master:~$ sudo firewall-cmd --permanent --add-port=10255/tcp
Warning: ALREADY_ENABLED: 10255:tcp
success
Master@Master:~$ sudo firewall-cmd --reload
success
Master@Master:~$
```

Enter the following commands on each worker node:

```
$ sudo firewall-cmd --permanent --add-port=10251/tcp
$ sudo firewall-cmd --permanent --add-port=10255/tcp
$ firewall-cmd --reload
```

7.2 Update Iptables Settings

To ensure packets are properly processed by IP tables during filtering and port forwarding. Set the net.bridge.bridge-nf-call-iptables to '1' in your sysctl config file.

```
$ cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
```

```
$ sysctl --system
```

```
root@Master:~# cat <<EOF > /etc/sysctl.d/k8s.conf
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> EOF
root@Master:~# sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
* Applying /etc/sysctl.d/10-link-restrictions.conf ...
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/10-lxd-inotify.conf ...
fs.inotify.max_user_instances = 1024
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
kernel.sysrq = 176
* Applying /etc/sysctl.d/10-network-security.conf ...
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.tcp_syncookies = 1
* Applying /etc/sysctl.d/10-ptrace.conf ...
kernel.yama.ptrace_scope = 1
* Applying /etc/sysctl.d/10-zeropage.conf ...
vm.mmap_min_addr = 65536
* Applying /usr/lib/sysctl.d/50-default.conf ...
net.ipv4.conf.all.promote_secondaries = 1
net.core.default_qdisc = fq_codel
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.default.use_tempaddr = 0
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
* Applying /etc/sysctl.conf ...
```

7.3 Disable SELinux

SELinux needs to be set to permissive mode. Because containers need to access the host filesystem in most of the times. If you, set to permissive mode it will effectively disables its security functions.

Use the following command to disable SELinux:

```
$ sudo setenforce 0
$ sudo sed -i ‘s/^SELINUX=enforcing$/SELINUX=permissive/’ /etc/selinux/config
```

7.4 Disable SWAP

To work kubelet properly we need to disable SWAP

To disable SWAP use the following commands.

```
$ sudo sed -i '/swap/d' /etc/fstab
$ sudo swapoff -a
```

8 KUBEADM TO CREATE AND INITIALISE A CLUSTER

1. Initialising the control-plane node run the below command on the **(master node)**

```
$ kubeadm init --kubernetes-version=${KUBE_VERSION}
```

```
root@Master:~# kubeadm init --kubernetes-version=${KUBE_VERSION}
[init] Using Kubernetes version: v1.21.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local master] and IPs
 [10.96.0.1 10.0.0.4]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and IPs [10.0.0.4 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs [10.0.0.4 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
```

2. If cluster initialisation has succeeded you will see a cluster join command. Copy and save that for future reference. This command would be used by the worker nodes to join the cluster

```
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.0.0.4:6443 --token 40tptl.10eqglw6k4i3zrrq \
        --discovery-token-ca-cert-hash sha256:aab219d719ae51e847df39df3e0344462224e8162aaf91ae07f30fa8379b6278
root@Master:~#
```

9 USING KUBEADM TO JOIN WORKER NODES TO THE CLUSTER

1. Run the above received kubeadm join command (on both the **worker nodes**)

Note: *This is above cluster command, you will get your command in your cluster so use that command not this command*

```
$ kubeadm join 10.0.0.4:6443 --token 40tptl.10eqglw6k4i3zrrq \
        --discovery-token-ca-cert-hash
sha256:aab219d719ae51e847df39df3e0344462224e8162aaf91ae07f30fa8379b6278
```

```
root@worker01:~# kubeadm join 10.0.0.4:6443 --token 40tptl.10eqglw6k4i3zrrq \
>         --discovery-token-ca-cert-hash sha256:aab219d719ae51e847df39df3e0344462224e8162aaf91ae07f30fa8379b6278
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@worker01:~#
```

Note: (Optional Step) If you forgot to add Save these tokens then you can always create a new token then use them to join Worker node

```
$ kubeadm token create --print-join-command
```

```
root@master:/home# kubeadm token create --print-join-command
W0807 11:35:35.839914   16801 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubel
et.config.k8s.io kubeproxy.config.k8s.io]
kubeadm join 10.0.4.4:6443 --token 8n46zi.u0vwo5h2yia42c83     --discovery-token-ca-cert-hash sha256:776e8fd4f4b456fa3b5
c3aaa6b78180df67c545309fe816fafdca7decba74a3f
root@master:/home#
```

Note: Everytime you are starting the Master you have to set these Enevironment Variables

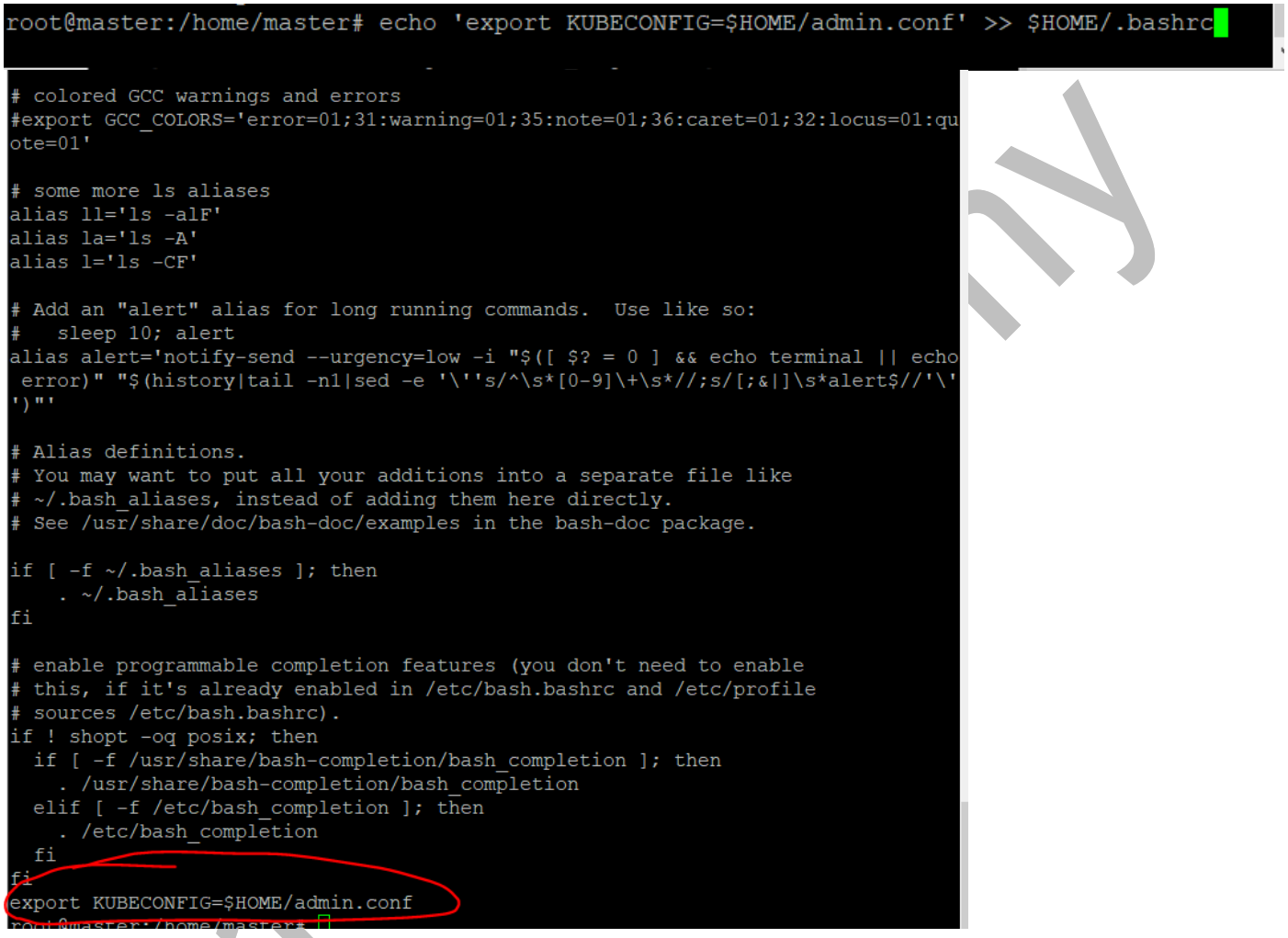
2. To start using the cluster set the environment variable on the *(master node)*

```
$ cp /etc/kubernetes/admin.conf $HOME/
$ chown $(id -u):$(id -g) $HOME/admin.conf
$ export KUBECONFIG=$HOME/admin.conf
```

```
root@kubeadm-master:/home/ubuntu# cp /etc/kubernetes/admin.conf $HOME/
root@kubeadm-master:/home/ubuntu# chown $(id -u):$(id -g) $HOME/admin.conf
root@kubeadm-master:/home/ubuntu# export KUBECONFIG=$HOME/admin.conf
root@kubeadm-master:/home/ubuntu#
```

Note: (Optional Step) To permanently set this environment variable, add it to your .bashrc file in your Root user.

```
$ echo 'export KUBECONFIG=$HOME/admin.conf' >> $HOME/.bashrc
```



3. Verify the cluster status by executing kubectl command on the master node *(You should see 3 machines here)*

```
$ kubectl get nodes
```

```
root@Master:~# kubectl get nodes
NAME       STATUS     ROLES                  AGE     VERSION
master     NotReady   control-plane,master   8m59s   v1.21.0
worker01   NotReady   <none>                 3m41s   v1.21.0
root@Master:~#
```

4. Install CNI so that pods can communicate across nodes and also Cluster DNS to start functioning. Apply weave CNI (Container Network Interface) on the master node

```
$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

```
root@kubeadm-master:/home/ubuntu# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n
')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu#
```

5. Wait for few minutes and verify the cluster status by executing kubectl command on the master node and see that nodes come to ready state *(You should see 3 machines here)*

```
$ kubectl get nodes
```

```
root@Master:~# kubectl get nodes
NAME       STATUS   ROLES                  AGE    VERSION
master     Ready    control-plane,master   13m    v1.21.0
worker01   Ready    <none>                 8m2s   v1.21.0
root@Master:~#
```

6. Verify the status of the system pods like coredns, weave-net, kube-proxy and all other master node system processes

```
$ kubectl get pods -n kube-system
```

```
root@Master:~# kubectl get pods -n kube-system
NAME                              READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-8jkhm          1/1     Running   0          14m
coredns-558bd4d5db-v7w6n          1/1     Running   0          14m
etcd-master                       1/1     Running   0          14m
kube-apiserver-master             1/1     Running   1          14m
kube-controller-manager-master    1/1     Running   0          14m
kube-proxy-6qhm4                  1/1     Running   0          8m56s
kube-proxy-xgdkm                  1/1     Running   0          14m
kube-scheduler-master             1/1     Running   0          14m
weave-net-9n2tz                   2/2     Running   1          2m56s
weave-net-t6gb6                   2/2     Running   1          2m56s
root@Master:~#
```

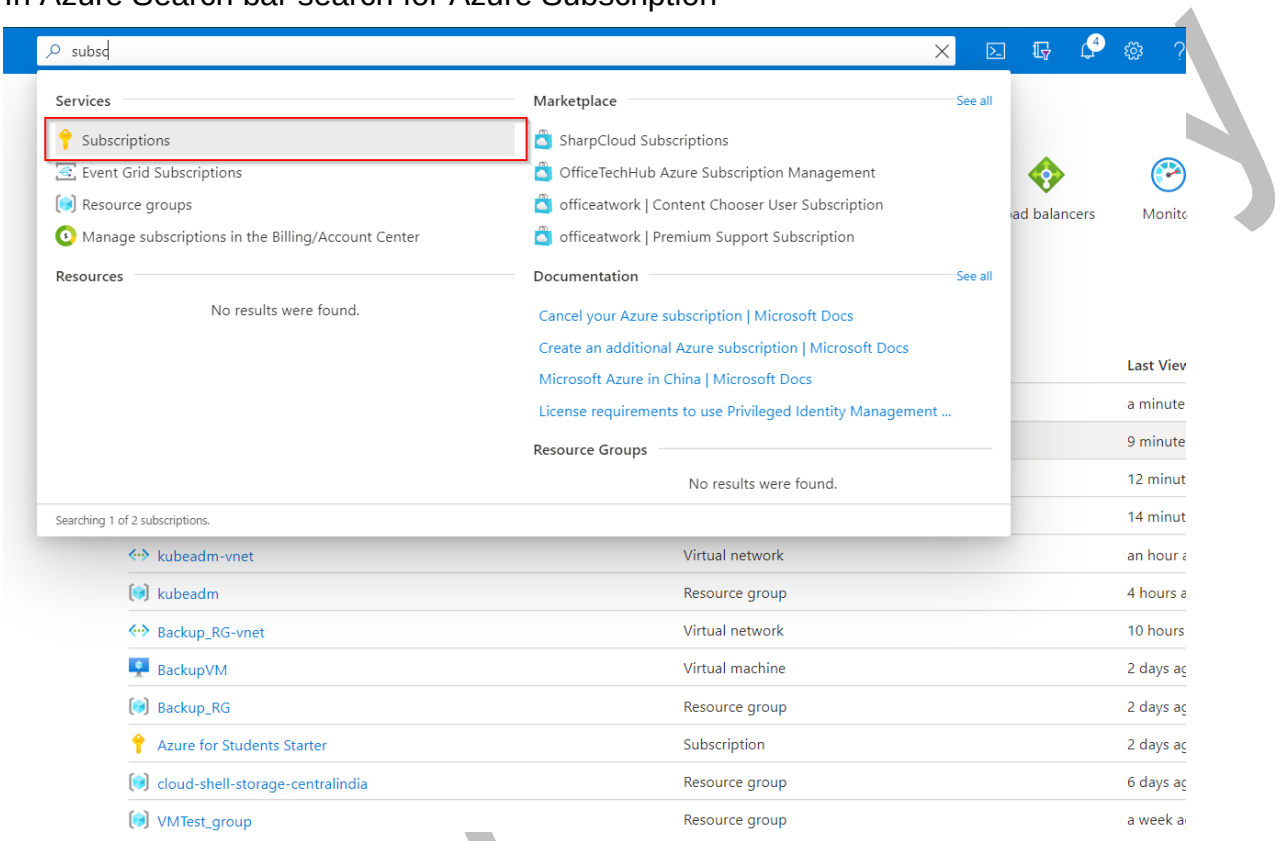
7. Git clone the code files to be used for further labs

```
$ git clone https://github.com/k21academyuk/Kubernetes
$ cd Kubernetes
```

10 (OPTIONAL) UNSUBSCRIBE PAY AS YOU GO IN AZURE

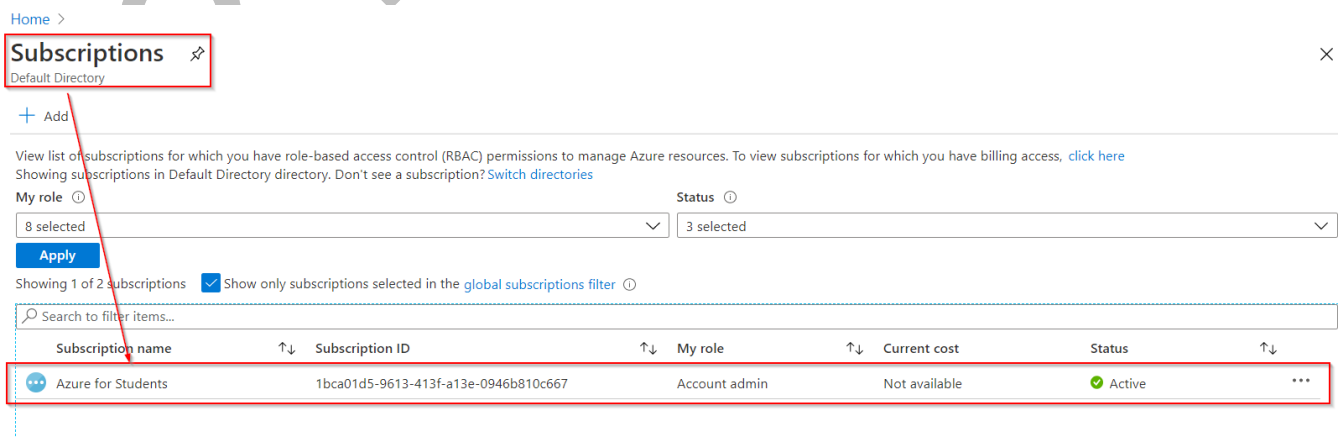
If you already upgarded to pay as you go then you can unsubscribe/deactivate the pay as you go and use delete all the resources.

1. In Azure Search bar search for Azure Subscription

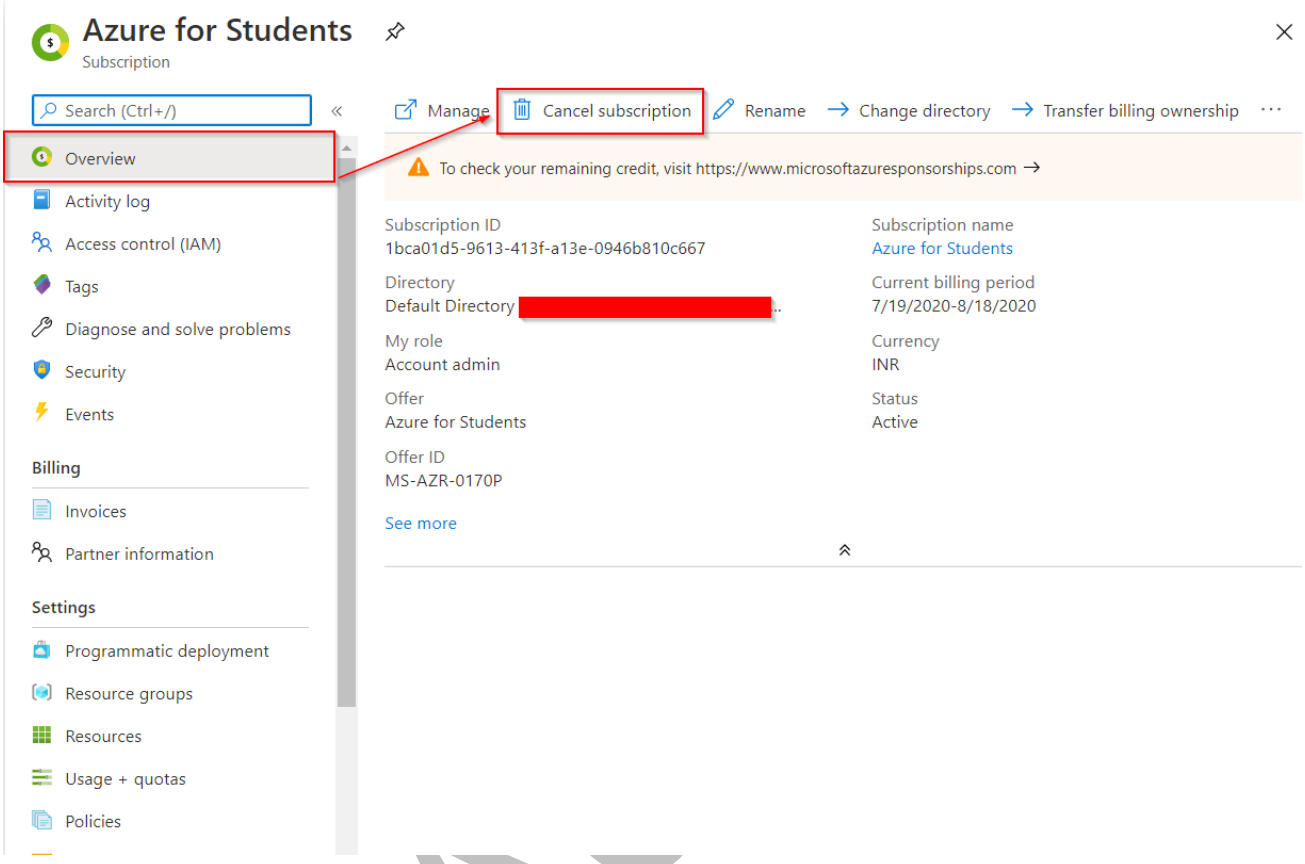


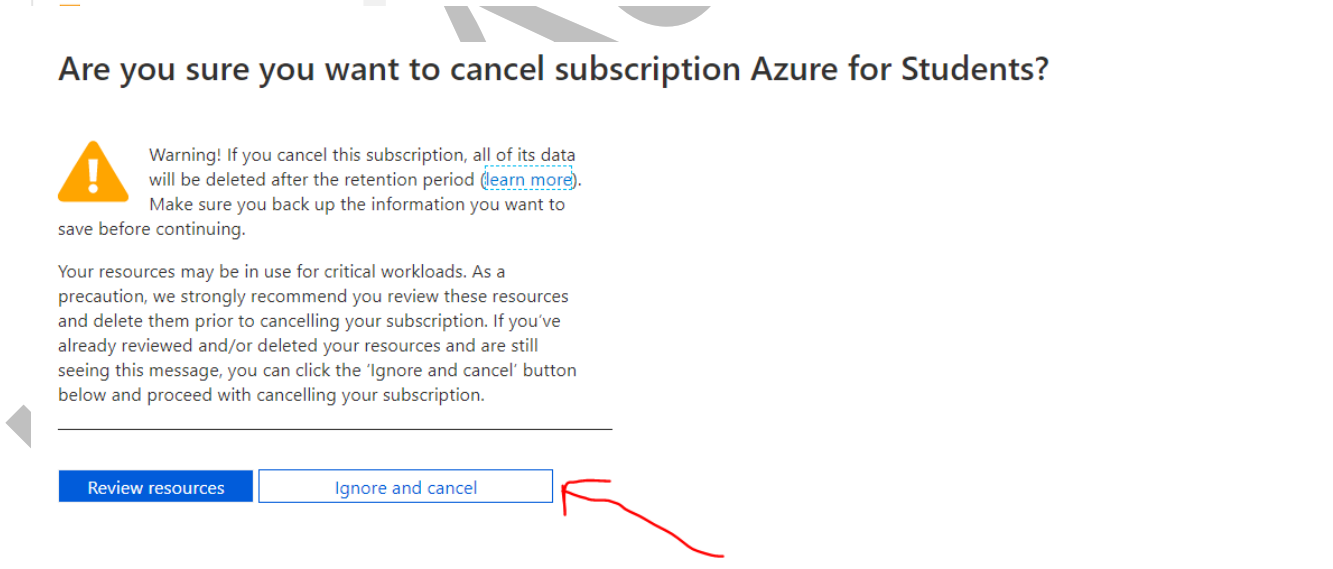
Note: In screenshots the account is Student account not as paid account but screens are same

2. Select the subscription that you want to cancel.



3. Select **Overview**, and then select Cancel subscription.





4. Now please create a new account and you can continue without pay-as-you go. Follow Create Azure account guide to create an account

5. FAQ

What happens after I cancel my subscription?

After you cancel, billing is stopped immediately. However, it can take up to 10 minutes for the cancellation to show in the portal. If you cancel in the middle of a billing period, we send the final invoice on your typical invoice date after the period ends.

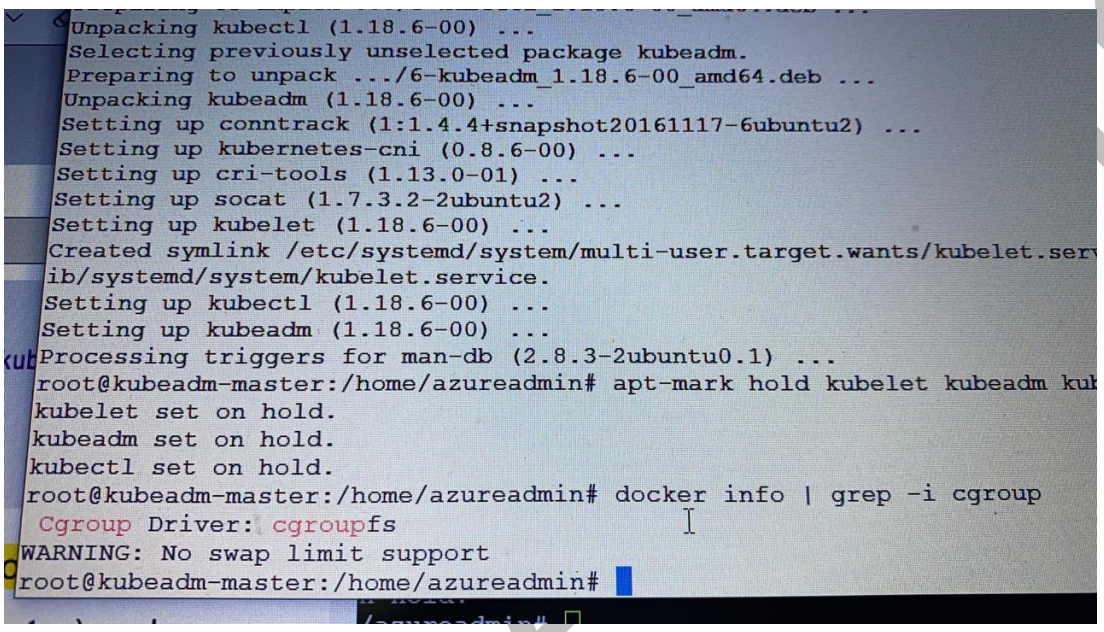
After you cancel, your services are disabled. That means your virtual machines are de-allocated, temporary IP addresses are freed, and storage is read-only.

After your subscription is canceled, Microsoft waits 30 - 90 days before permanently deleting your data in case you need to access it or you change your mind. We don't charge you for retaining the data. To learn more, see Microsoft Trust Center - How we manage your data.

11 TROUBLESHOOTING

11.1 Getting Warning while Configure cgroup driver used by kubelet on control-plane

Issue: Getting warning while like below Image



Fix: It's just a warning plz ignore this warning

11.2 Getting Error while Connecting node to Master node

Issue: getting error while running connecting node to master node command

```
root@master:~# kubeadm join 10.0.4.4:6443 --token x0ov1c.bu8xnx9154fe77on \
> --discovery-token-ca-cert-hash sha256:776e8fd4f4b456fa3b5c3aaa6b78180df67c545309fe816fafdca7decba74a3f
W0728 10:50:03.928021   44127 join.go:346] [preflight] WARNING: JoinControlPane.controlPlane settings will be ignored when control-plane flag is not set.
[preflight] Running pre-flight checks
        [WARNING Service-Docker]: docker service is not enabled, please run 'systemctl enable docker.service'
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kuber
.io/docs/setup/cri/
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR DirAvailable--etc-kubernetes-manifests]: /etc/kubernetes/manifests is not empty
        [ERROR FileAvailable--etc-kubernetes-kubelet.conf]: /etc/kubernetes/kubelet.conf already exists
        [ERROR Port-10250]: Port 10250 is in use
        [ERROR FileAvailable--etc-kubernetes-pki-ca.crt]: /etc/kubernetes/pki/ca.crt already exists
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Reason: If you are getting error like above then you are running connecting command in master node instead of worker node

Fix: Please run above Command in Worker Nodes.

11.3 Running kubectl get nodes and getting server localhost:8080

Issue: If you are running any command like kubectl get nodes and getting error like below

```
root@kubernete-master:/home/AzureUser# kubectl get nodes
The connection to the server localhost:8080 was refused - did you specify the
right host or port?
root@kubernete-master:/home/AzureUser# kubectl apply -f "
https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -
d '\n')"
The connection to the server localhost:8080 was refused - did you specify the
right host or port?
The connection to the server localhost:8080 was refused - did you specify the
right host or port?
                                                                    4:33 AM
```

Reason: Environment variable not set on the master node

Fix: set the environment variable on the master nod.

2. To start using the cluster set the environment variable on the master node

```
$ cp /etc/kubernetes/admin.conf $HOME/
$ chown $(id -u):$(id -g) $HOME/admin.conf
$ export KUBECONFIG=$HOME/admin.conf
```

```
root@kubeadm-master:/home/ubuntu# cp /etc/kubernetes/admin.conf $HOME/
root@kubeadm-master:/home/ubuntu# chown $(id -u):$(id -g) $HOME/admin.conf
root@kubeadm-master:/home/ubuntu# export KUBECONFIG=$HOME/admin.conf
root@kubeadm-master:/home/ubuntu#
```

11.4 Error when Running Kubeadm init Command

Issue: If you are running kubeadm init command and getting error like below

```
root@master:/home/ubuntu# kubeadm init
[init] Using Kubernetes version: v1.20.4
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at
https://kubernetes.io/docs/setup/cri/
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR Port-6443]: Port 6443 is in use
        [ERROR Port-10259]: Port 10259 is in use
        [ERROR Port-10257]: Port 10257 is in use
        [ERROR FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already exists
        [ERROR FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-manager.yaml already exists
        [ERROR FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already exists
        [ERROR FileAvailable--etc-kubernetes-manifests-etcd.yaml]: /etc/kubernetes/manifests/etcd.yaml already exists
        [ERROR Port-10250]: Port 10250 is in use
        [ERROR Port-2379]: Port 2379 is in use
        [ERROR Port-2380]: Port 2380 is in use
        [ERROR DirAvailable--var-lib-etcd]: /var/lib/etcd is not empty
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Reason: Because you are running kubeadm init command again

Fix: please check if kubernetes cluster running or not already by

```
kubectl get nodes
```

if its not running please run below command and run **kubeadm init** command again.

```
kubeadm reset
```

11.5 Using Expired Token When Joining Node

Issue: Can't join Worker Node to master node using previous created tokens.

Reason: Tokens created init command by default expires after the 24 hours so you have to create a new token to join a new worker node to master node.

Fix: Create new tokens using below command and use new tokens.

```
$ kubeadm token create --print-join-command
```

11.6 Error when Running Kubeadm init Command

When you are doing Optional guide [Bonus] [Optional] Connect Windows Worker Node to Ubuntu Master Node we have installed **Flannel CNI** so due to this this is creating issue.

Issue: two CNI plugin installed weave & flannel and flannel is not working.

```
sc = failed to set up sandbox container "060839caaab9b17e26aa72487610e395cedeca6b63a07afdc90e
o set up pod "nginx_default" network: open /run/flannel/subnet.env: no such file or directory
root@master:~# kubectl get pods -n kube-system
NAME                              READY   STATUS             RESTARTS   AGE
coredns-74ff55c5b-hlgs5           0/1     Completed          0          4d23h
coredns-74ff55c5b-ntncp           0/1     Completed          0          4d23h
etcd-master                       1/1     Running            5          4d23h
kube-apiserver-master             1/1     Running            6          4d23h
kube-controller-manager-master    1/1     Running            5          4d23h
kube-flannel-ds-2vdwf             0/1     CrashLoopBackOff   180        4d22h
kube-flannel-ds-56wdj             0/1     CrashLoopBackOff   181        4d22h
kube-flannel-ds-t6cvg             0/1     CrashLoopBackOff   180        4d22h
kube-proxy-78qj2                  1/1     Running            5          4d23h
kube-proxy-tq8zk                  1/1     Running            5          4d23h
kube-proxy-xr2mp                  1/1     Running            5          4d23h
kube-scheduler-master             1/1     Running            5          4d23h
weave-net-2wr5q                   2/2     Running            11         4d22h
weave-net-xch7t                   2/2     Running            11         4d22h
weave-net-zx2b9                   2/2     Running            11         4d22h
root@master:~# kubectl describe pod kube-flannel-ds-2vdwf -n kube-system
Name:                 kube-flannel-ds-2vdwf
Namespace:            kube-system
```

```
root@master:/home/ubuntu# kubectl get pods --output=wide
NAME                    READY   STATUS              RESTARTS   AGE     IP       NODE        NOMINATED NOD
E   READINESS GATES
nginx                   0/1     ContainerCreating   0          2m47s   <none>   worker-01   <none>
    <none>
test-replicaset-7m8fl   0/1     ContainerCreating   0          34m     <none>   worker-02   <none>
    <none>
test-replicaset-rtqpg   0/1     ContainerCreating   0          34m     <none>   worker-02   <none>
    <none>
```

Reason: flannel CNI plug-in not working so pod not scheduling on the worker nodes because pods are using flannel CNI

Fix: Remove Flannel from the cluster from the master node

```
$ kubectl delete -f kube-flannel.yml
```

```
$ kubectl delete -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

Install Weave again

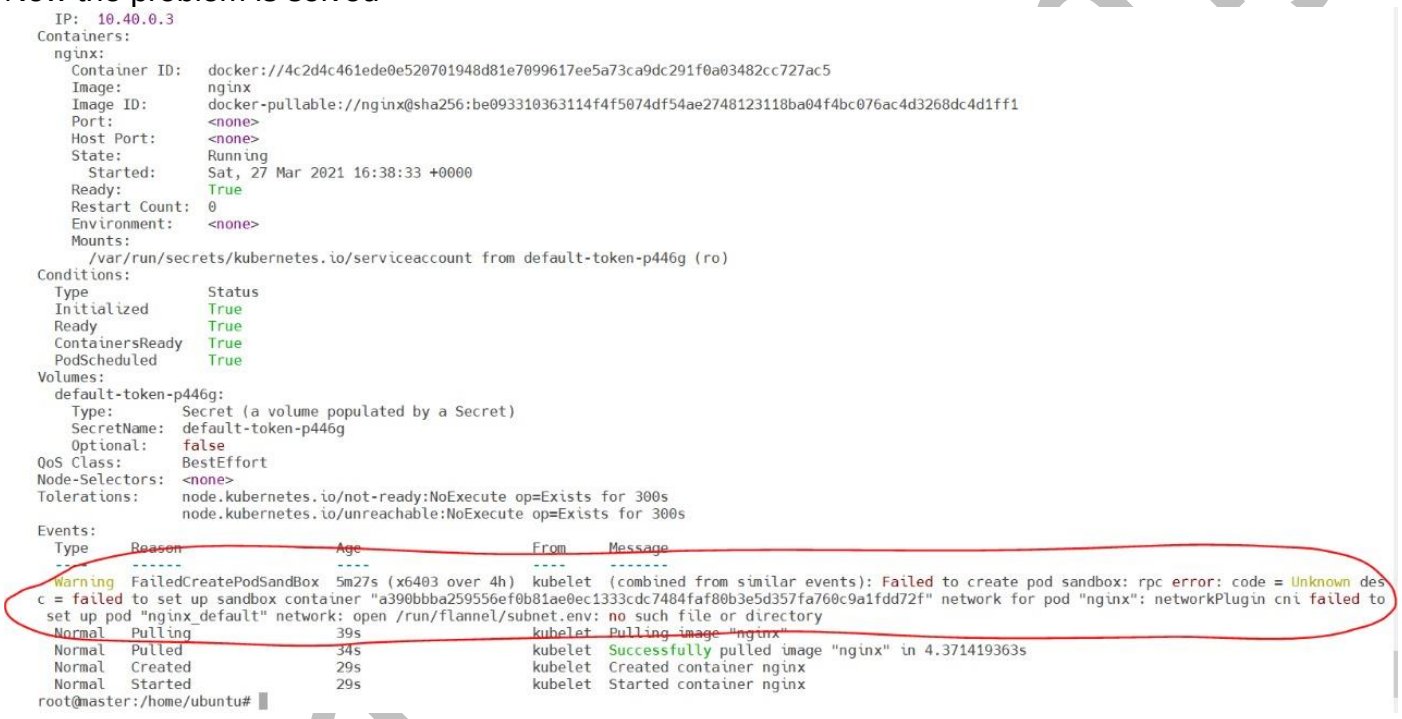
```
$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

Perfrom below stepson all the nodes master and worker

```
$ rm /etc/cni/net.d/10-flannel.conflist

$ systemctl restart kubelet
```

Now the problem is solved



11.7 Unable to Ping Master node to Worker after Installing Docker

Issue: When we create 3 machine for Kubernetes by default they can ping to each other but after isntalling the docker on the master machine unable to ping machies.

```
root@master:~# ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4) 56(84) bytes of data.
64 bytes from 172.17.0.4: icmp_seq=2 ttl=64 time=44.0 ms
64 bytes from 172.17.0.4: icmp_seq=3 ttl=64 time=33.0 ms
64 bytes from 172.17.0.4: icmp_seq=4 ttl=64 time=27.0 ms
^C64 bytes from 172.17.0.4: icmp_seq=5 ttl=64 time=27.5 ms
64 bytes from 172.17.0.4: icmp_seq=6 ttl=64 time=27.3 ms
64 bytes from 172.17.0.4: icmp_seq=7 ttl=64 time=29.9 ms
^C
--- 172.17.0.4 ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time 6024ms
rtt min/avg/max/mdev = 27.051/31.509/44.047/5.978 ms
root@master:~# ping 172.17.0.5
PING 172.17.0.5 (172.17.0.5) 56(84) bytes of data.
64 bytes from 172.17.0.5: icmp_seq=1 ttl=64 time=32.8 ms
64 bytes from 172.17.0.5: icmp_seq=2 ttl=64 time=32.4 ms
64 bytes from 172.17.0.5: icmp_seq=3 ttl=64 time=32.4 ms
^C
--- 172.17.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 32.492/32.625/32.886/0.278 ms
root@master:~# ping google.com
PING google.com (172.217.12.238) 56(84) bytes of data.
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=1 ttl=111 time=5.34 ms
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=2 ttl=111 time=5.22 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 5.228/5.286/5.345/0.093 ms
```

```
PING 172.16.0.4 (172.16.0.4) 56(84) bytes of data.
64 bytes from 172.16.0.4: icmp_seq=1 ttl=64 time=27.8 ms
64 bytes from 172.16.0.4: icmp_seq=2 ttl=64 time=27.1 ms
^C
--- 172.16.0.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 27.134/27.470/27.806/0.336 ms
root@Worker-1:~# ping 172.17.0.5
PING 172.17.0.5 (172.17.0.5) 56(84) bytes of data.
64 bytes from 172.17.0.5: icmp_seq=1 ttl=64 time=2.34 ms
64 bytes from 172.17.0.5: icmp_seq=2 ttl=64 time=1.48 ms
64 bytes from 172.17.0.5: icmp_seq=3 ttl=64 time=1.31 ms
^C
--- 172.17.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.313/1.715/2.345/0.452 ms
root@Worker-1:~# ping google.com
PING google.com (142.250.113.139) 56(84) bytes of data.
^C64 bytes from 142.250.113.139: icmp_seq=1 ttl=108 time=7.92 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.926/7.926/7.926/0.000 ms
root@Worker-1:~# ping google.com
PING google.com (142.250.113.101) 56(84) bytes of data.
64 bytes from 142.250.113.101: icmp_seq=1 ttl=105 time=9.32 ms
64 bytes from 142.250.113.101: icmp_seq=2 ttl=105 time=9.39 ms
64 bytes from 142.250.113.101: icmp_seq=3 ttl=105 time=9.35 ms
64 bytes from 142.250.113.101: icmp_seq=4 ttl=105 time=9.36 ms
64 bytes from 142.250.113.101: icmp_seq=5 ttl=105 time=9.35 ms
64 bytes from 142.250.113.101: icmp_seq=6 ttl=105 time=9.32 ms
64 bytes from 142.250.113.101: icmp_seq=7 ttl=105 time=9.68 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 15027ms
rtt min/avg/max/mdev = 9.320/9.398/9.685/0.173 ms
root@Worker-1:~#
```

```
ubuntu@Worker-2:~$ sudo -i
root@Worker-2:~# ping 172.16.0.4
PING 172.16.0.4 (172.16.0.4) 56(84) bytes of data.
64 bytes from 172.16.0.4: icmp_seq=1 ttl=64 time=33.0 ms
64 bytes from 172.16.0.4: icmp_seq=2 ttl=64 time=32.6 ms
^C
--- 172.16.0.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 32.641/32.821/33.001/0.180 ms
root@Worker-2:~# ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4) 56(84) bytes of data.
64 bytes from 172.17.0.4: icmp_seq=1 ttl=64 time=1.37 ms
64 bytes from 172.17.0.4: icmp_seq=2 ttl=64 time=1.37 ms
64 bytes from 172.17.0.4: icmp_seq=3 ttl=64 time=1.38 ms
^C
--- 172.17.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.374/1.378/1.388/0.043 ms
root@Worker-2:~# ping google.com
PING google.com (172.217.14.174) 56(84) bytes of data.
64 bytes from dfw28s22-in-f14.1e100.net (172.217.14.174): icmp_seq=1 ttl=116 time=8.32 ms
64 bytes from dfw28s22-in-f14.1e100.net (172.217.14.174): icmp_seq=2 ttl=116 time=8.47 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 8.328/8.399/8.470/0.071 ms
root@Worker-2:~#
```

After installing the docker on master machine

```
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.0.0~rc93-0ubuntu1~18.04.2_amd64.deb ...
Unpacking runc (1.0.0~rc93-0ubuntu1~18.04.2) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.4.4-0ubuntu1~18.04.2_amd64.deb ...
Unpacking containerd (1.4.4-0ubuntu1~18.04.2) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../4-docker.io_20.10.2-0ubuntu1~18.04.2_amd64.deb ...
Unpacking docker.io (20.10.2-0ubuntu1~18.04.2) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../5-ubuntu-fan_0.12.10_all.deb ...
Unpacking ubuntu-fan (0.12.10) ...
Setting up runc (1.0.0~rc93-0ubuntu1~18.04.2) ...
Setting up containerd (1.4.4-0ubuntu1~18.04.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service â /lib/systemd/system/containerd.service.
Setting up bridge-utils (1.5-15ubuntu1) ...
Setting up ubuntu-fan (0.12.10) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service â /lib/systemd/system/ubuntu-fan.service.
Setting up pigz (2.4-1) ...
Setting up docker.io (20.10.2-0ubuntu1~18.04.2) ...
Adding group `docker' (GID 116) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service â /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket â /lib/systemd/system/docker.socket.
Processing triggers for systemd (237-3ubuntu10.47) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
root@master:~# ping 172.17.0.5
PING 172.17.0.5 (172.17.0.5) 56(84) bytes of data.
From 172.17.0.1 icmp_seq=1 Destination Host Unreachable
From 172.17.0.1 icmp_seq=2 Destination Host Unreachable
From 172.17.0.1 icmp_seq=3 Destination Host Unreachable
^C
--- 172.17.0.5 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5099ms
pipe 4
root@master:~# apt-get update && apt install -y docker.io
```

```
root@master:~# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:65:31:f0:20  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.0.4  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::20d:3aff:fe7b:c854  prefixlen 64  scopeid 0x20<link>
        ether 00:0d:3a:7b:c8:54  txqueuelen 1000  (Ethernet)
        RX packets 154141  bytes 193565826 (193.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40858  bytes 7940474 (7.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4539  bytes 566139 (566.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4539  bytes 566139 (566.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
root@Worker-1:~# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.18.0.1  netmask 255.255.0.0  broadcast 172.18.255.255
        ether 02:42:fc:c6:14:a1  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.4  netmask 255.255.255.0  broadcast 172.17.0.255
        inet6 fe80::20d:3aff:fe5d:e47c  prefixlen 64  scopeid 0x20<link
        ether 00:0d:3a:5d:e4:7c  txqueuelen 1000  (Ethernet)
        RX packets 96584  bytes 115191527 (115.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32918  bytes 7056836 (7.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 6385  bytes 687091 (687.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6385  bytes 687091 (687.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Reason: When we isntall docker on the machine by default the docker0 takes default IP that is 172.17.0.1/16 so this IP range is conflicting with the other two worker nodes private IP that is 172.17.0.4 and 172.17.0.5 that's why when we are ping to master to worker node it pinging using the docker0 instead of host because both in same IP range.

```
ubuntu@master:~$ ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4) 56(84) bytes of data.
From 172.17.0.1 icmp_seq=1 Destination Host Unreachable
From 172.17.0.1 icmp_seq=2 Destination Host Unreachable
From 172.17.0.1 icmp_seq=3 Destination Host Unreachable
From 172.17.0.1 icmp_seq=4 Destination Host Unreachable
From 172.17.0.1 icmp_seq=5 Destination Host Unreachable
From 172.17.0.1 icmp_seq=6 Destination Host Unreachable
^C
--- 172.17.0.4 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6145ms
pipe 4
ubuntu@master:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:b9:ba:9a:37  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.0.4  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::20d:3aff:fe7b:c854  prefixlen 64  scopeid 0x20<link>
        ether 00:0d:3a:7b:c8:54  txqueuelen 1000  (Ethernet)
        RX packets 88661  bytes 111897087 (111.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22430  bytes 4663415 (4.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2888  bytes 355031 (355.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2888  bytes 355031 (355.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@master:~$
```

Fix: First, explicitly set the address of docker0 in your **/etc/docker/daemon.json** Use any network that doesn't conflict your worker node networks.

Reference: https://superuser.com/questions/1336567/installing-docker-ce-in-ubuntu-18-04-breaks-internet-connectivity-of-host/1380836#1380836?newreg=74f46627c99344e2b382b6ab3efd49ed

If **/etc/docker/daemon.json not present then create new file**

```
$ vi /etc/docker/daemon.json
```

Add any Ip range in **bip field**

```
{
  "bip": "172.31.0.1/16"
}
```

```
{
 "bip": "172.31.0.1/16"
}

~
~
~
```

Now restart the docker

```
$ service docker restart
```

```
root@master:~# vim /etc/docker/daemon.json
root@master:~# service docker restart
```

Now the Ip of docker0 is changed now try to ping again from master to worker machine. Problem is resolved.

```
root@master:~# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.31.0.1  netmask 255.255.0.0  broadcast 172.31.255.255
        ether 02:42:65:31:f0:20  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.0.4  netmask 255.255.255.0  broadcast 172.16.0.255
        inet6 fe80::20d:3aff:fe7b:c854  prefixlen 64  scopeid 0x20<link>
        ether 00:0d:3a:7b:c8:54  txqueuelen 1000  (Ethernet)
        RX packets 158373  bytes 195609150 (195.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45216  bytes 8933809 (8.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 5129  bytes 641702 (641.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5129  bytes 641702 (641.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@master:~# ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4) 56(84) bytes of data.
64 bytes from 172.17.0.4: icmp_seq=1 ttl=64 time=27.7 ms
64 bytes from 172.17.0.4: icmp_seq=2 ttl=64 time=27.3 ms
64 bytes from 172.17.0.4: icmp_seq=3 ttl=64 time=27.2 ms
^C
--- 172.17.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 27.291/27.452/27.704/0.180 ms
root@master:~#
```

12 SUMMARY

In this guide we Covered:

- Bootstrap A Kubernetes Cluster Using Kubeadm