Experiment – 10

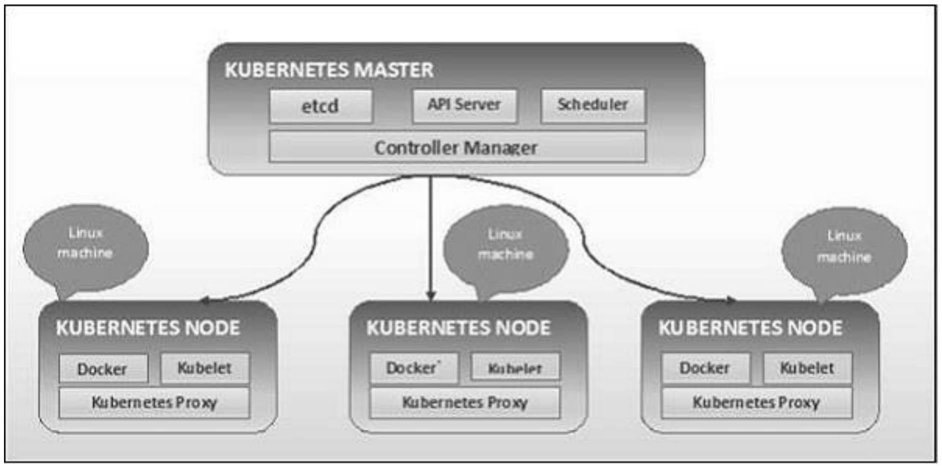
NAME : Ronak Surve

ROLL NO : 64

YEAR : 2022-2023

SUBJECT NAME : CLOUD COMPUTING

To study and implement container orchestration using Kubernetes.	
Learning Objective:	To study and implement container orchestration using Kubernetes
Learning Outcome:	Students will be able to understand the container orchestration using kubernetes on local machine or cloud system
Course Outcome:	CSL605.6
Program Outcome:	1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. 2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. 3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. 4. Conduct investigations of complex problems: Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

Bloom's Taxonomy Level:	Analysis, Apply
Theory:	**Explain Concept and feature of Kubernetes in details** Kubernetes in an open source container management tool hosted by Cloud Native Computing Foundation (CNCF). This is also known as the enhanced version of Borg which was developed at Google to manage both long running processes and batch jobs, which was earlier handled by separate systems. Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container centric infrastructure. the basic architecture of Kubernetes. Kubernetes - Cluster Architecture As seen in the following diagram, Kubernetes follows client-server architecture. Wherein, we have master installed on one machine and the node on separate Linux machines.  KUBERNETES MASTER etcd API Server Scheduler Controller Manager Linux machine Linux machine Linux machine KUBERNETES NODE Docker Kubelet Kubernetes Proxy KUBERNETES NODE Docker Kubelet Kubernetes Proxy KUBERNETES NODE Docker Kubelet Kubernetes Proxy The key components of master and node are defined in the following section. Kubernetes - Master Machine Components Following are the components of Kubernetes Master Machine.

etcd

It stores the configuration information which can be used by each of the nodes in the cluster. It is a high availability key value store that can be distributed among multiple nodes. It is accessible only by Kubernetes API server as it may have some sensitive information. It is a distributed key value Store which is accessible to all.

API Server

Kubernetes is an API server which provides all the operation on cluster using the API. API server implements an interface, which means different tools and libraries can readily communicate with it. **Kubeconfig** is a package along with the server side tools that can be used for communication. It exposes Kubernetes API.

Controller Manager

This component is responsible for most of the collectors that regulates the state of cluster and performs a task. In general, it can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server. It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state. The key controllers are replication controller, endpoint controller, namespace controller, and service account controller. The controller manager runs different kind of controllers to handle nodes, endpoints, etc.

Scheduler

This is one of the key components of Kubernetes master. It is a service in master responsible for distributing the workload. It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload. In other words, this is the mechanism responsible for allocating pods to available nodes. The scheduler is responsible for workload utilization and allocating pod to new node.

Kubernetes - Node Components

Following are the key components of Node server which are necessary to communicate with Kubernetes master.

Docker

The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment.

Kubelet Service

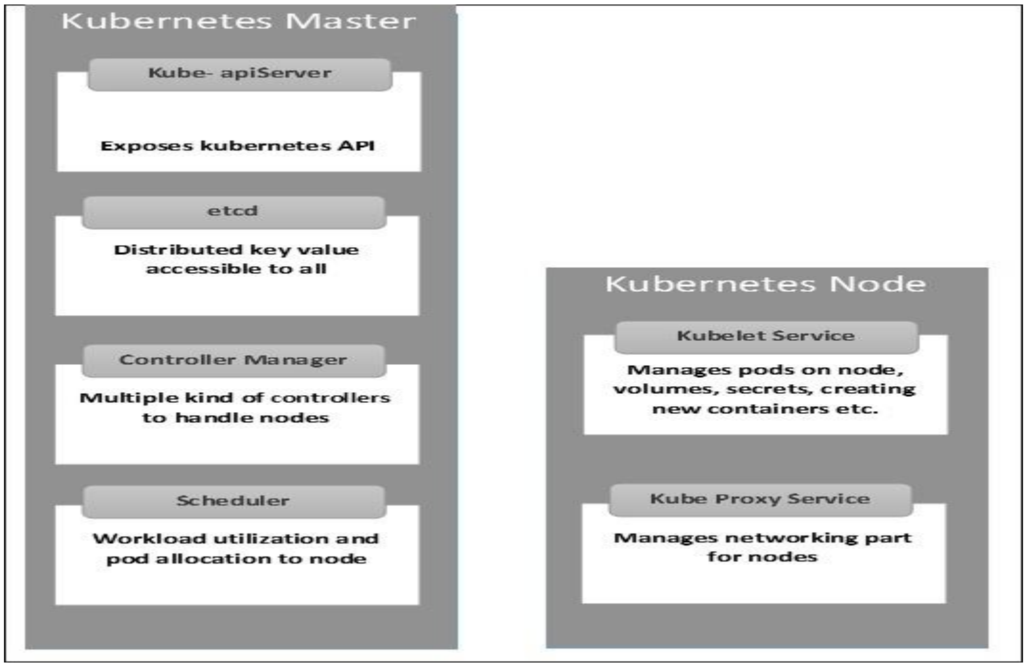
This is a small service in each node responsible for relaying information to and from control plane service. It interacts with **etcd** store to read configuration details and wright values. This communicates with the master component to receive commands and work. The **kubelet** process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

Kubernetes Proxy Service

This is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and is capable of performing primitive load balancing. It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well. It manages pods on node, volumes, secrets, creating new containers' health checkup, etc.

Kubernetes - Master and Node Structure

It is important to set up the Virtual Datacenter (vDC) before setting up Kubernetes. This can be considered as a set of machines where they can communicate with each other via the network. For hands-on approach, you can set up vDC on PROFITBRICKS **if you do not have a physical or cloud infrastructure set up.**

	Once the IaaS setup on any cloud is complete, you need to configure the **Master** and the **Node**. The following illustrations show the structure of Kubernetes Master and Node.  Kubernetes Master Kube- apiServer Exposes kubernetes API etcd Distributed key value accessible to all Controller Manager Multiple kind of controllers to handle nodes Scheduler Workload utilization and pod allocation to node Kubernetes Node Kubelet Service Manages pods on node, volumes, secrets, creating new containers etc. Kube Proxy Service Manages networking part for nodes **Kubernetes - Setup** **Features of Kubernetes** a. Continues development, integration and deployment b. Containerized infrastructure c. Application-centric management d. Auto-scalable infrastructure e. Environment consistency across development testing and production f. Loosely coupled infrastructure, where each component can act as a separate unit g. Higher density of resource utilization h. Predictable infrastructure which is going to be created
Procedure	To understand the steps to deploy Kubernetes Cluster on local systems, deploy applications on Kubernetes, creating a Service in Kubernetes, develop Kubernetes

	configuration files in YAML and creating a deployment in Kubernetes using YAML,
Steps	**Step 1 - Update Ubuntu** Always recommended updating the system packages. So let's go to the command: #sudo apt update Then type: #sudo apt upgrade **Step 2 - Install Docker** Kubernetes requires an existing Docker installation. Install Docker with the command: #sudo apt install docker.io Repeat the process on each server that will act as a node. Check the installation (and version) by entering the following: #docker —version **Step 3 - Start and Enable Docker** Set Docker to launch at boot by entering the following: #sudo systemctl enable docker #sudo systemctl start docker Verify Docker is running: #sudo systemctl status docker ● docker.service - Docker Application Container Engine Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled) Active: active (running) since Sat 2020-10-20 13:53:02 -03; 2h 54min ago TriggeredBy: ● docker.socket Docs: https://docs.docker.com

```
   Main PID: 2282 (dockerd)
      Tasks: 21
     Memory: 146.3M
     CGroup: /system.slice/docker.service
             └─2282 /usr/bin/dockerd -H fd:// --
containerd=/run/containerd/containerd.sock
```

To start Docker if it's not running:

```
#sudo systemctl start docker
```

Repeat on all the other nodes

.

Step 4 - Install Kubernetes

As we are downloading Kubernetes from a non-standard repository, it is essential to ensure that the software is authentic. This is done by adding a subscription key.

Enter the following to add a signing key in you on Ubuntu:

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

Then repeat the previous command to install the signing keys.

Repeat for each server node.

Step 5 - Add Software Repositories

Kubernetes is not included in the default repositories. To add them, enter the following:

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

Repeat for each server node.

Step 6 - Kubernetes Installation Tools

Kubernetes Admin or Kubeadm is a tool that helps initialize a cluster. Its fast-track setup by using community-sourced best practices. Kubelet is the work package, which runs on every node and starts containers. The tool gives you command-line access to clusters.

Install Kubernetes tools with the command:

```
#sudo apt-get install kubeadm kubelet kubectl
#sudo apt-mark hold kubeadm kubelet kubectl
```

Allow the process to complete.

Verify the installation with:

```
#kubeadm version
```

Repeat for each server node.

Step 7 - Kubernetes Deployment

Begin Kubernetes Deployment

Start by disabling the swap memory on each server:

```
#sudo swapoff –a
```

Step 8 - Assign Unique Hostname for Each Server Node

Decide which server to set as the master node. Then enter the command:

```
#sudo hostnamectl set-hostname master-node
```

Next, set a worker node hostname by entering the following on the worker server:

```
#sudo hostnamectl set-hostname w1
```

If you have additional worker nodes, use this process to set a unique hostname on each.

Step 9 - Initialize Kubernetes on Master Node

Switch to the master server node, and enter the following:

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

Once this command finishes, it will display a kubeadm join message at the end. Make a note of the whole entry. This will be used to join the worker nodes to the cluster.

Next, enter the following to create a directory for the cluster:

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

Step 10 - Deploy Pod Network to Cluster

A Pod Network is a way to allow communication between different nodes in the cluster. This tutorial uses the flannel virtual network.

Enter the following:

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

Allow the process to complete.

Verify that everything is running and communicating:

```
#
kubectl get pods --all-namespaces
```

Step 11 - Join the Worker Node to Cluster

As indicated in Step 7, you can enter the kubeadm join command on each worker node to connect it to the cluster.

Switch to the w1 system and enter the command you noted from Step 8:

```
#kubeadm join --discovery-token abcdef.1234567890abcdef --discovery token-ca-cert-hash sha256:1234..cdef 1.2.3.4:6443
```

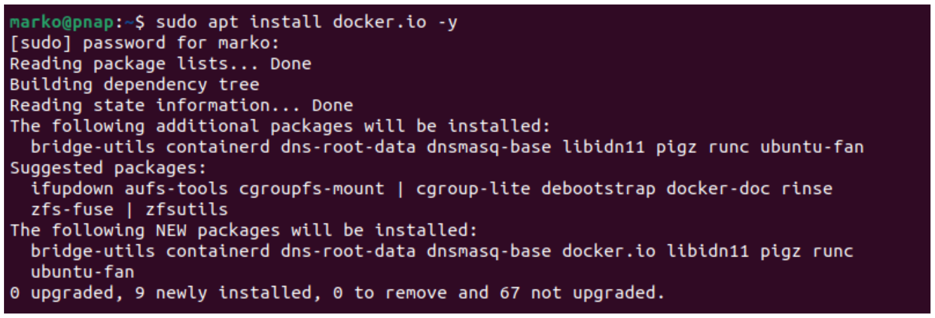
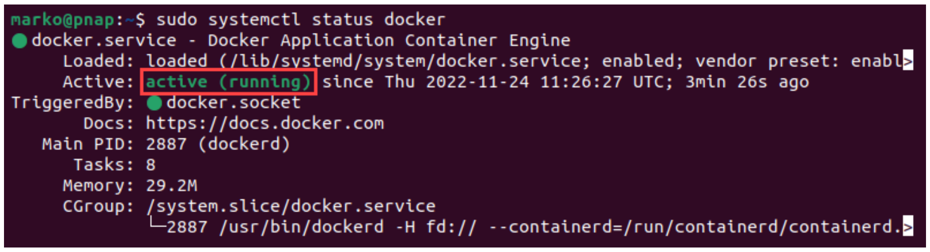
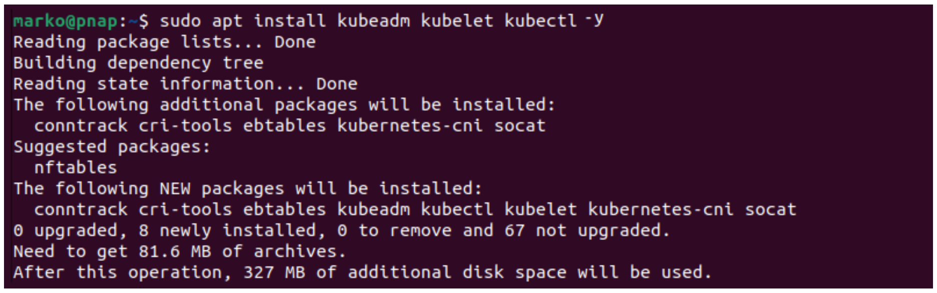
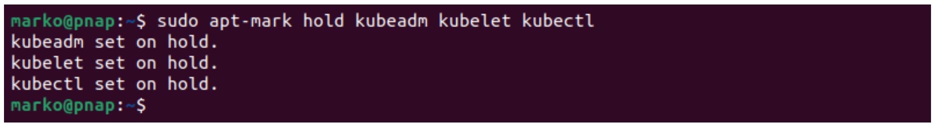
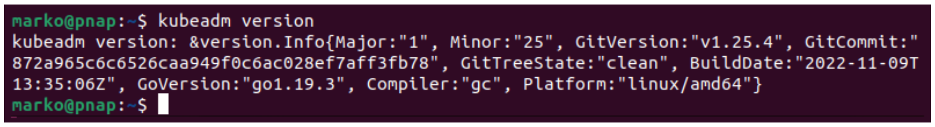
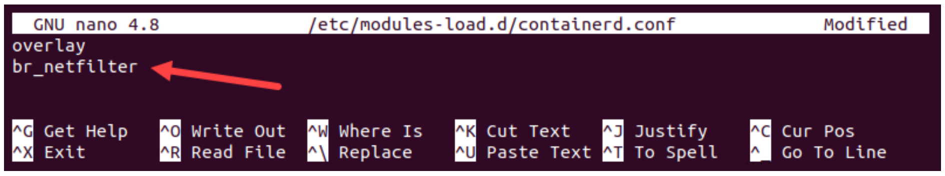
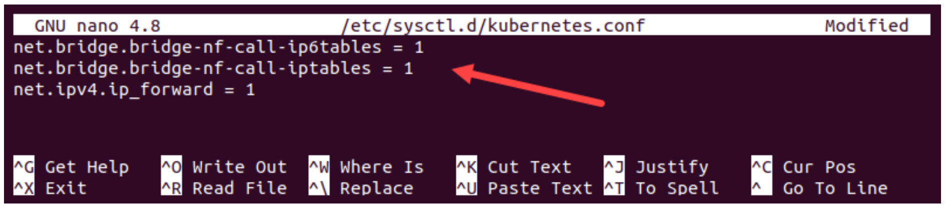
Replace the alphanumeric codes with those from your master server. Repeat for each worker node on the cluster. Wait a few minutes; then you can check the status of the nodes.

Switch to the master server, and enter:

```
#kubectl get nodes
```

The system should display the worker nodes that you joined to the cluster.

Node	**Hostname**	**IP Address**	**vCPUs**	**RAM (GB)**	**OS**
Master	master.letscloud.io	192.168.51.111	2	3.75	Ubuntu 20.04
W1	w1.letscloud.io	192.168.52.8	2	3.75	Ubuntu 20.04
W2	w2.letscloud.io	192.168.58.6	2	3.75	Ubuntu 20.04

Outcome :	      

```
marko@pnap:~$ sudo apt install docker.io -y
[sudo] password for marko:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse
  zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc
  ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 67 not upgraded.
```

```
marko@pnap:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabl>
     Active: active (running) since Thu 2022-11-24 11:26:27 UTC; 3min 26s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2887 (dockerd)
      Tasks: 8
     Memory: 29.2M
     CGroup: /system.slice/docker.service
             └─2887 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.>
```

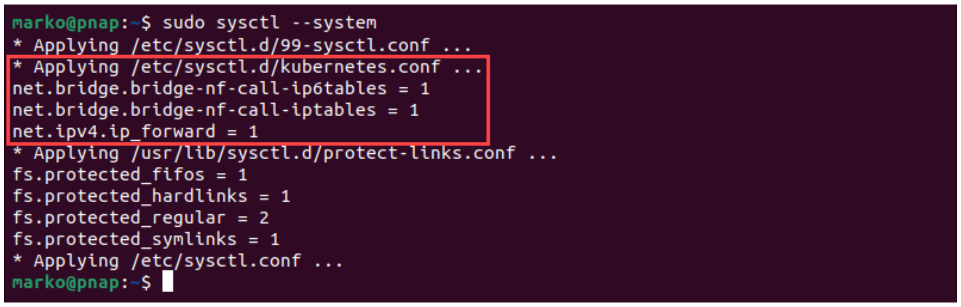
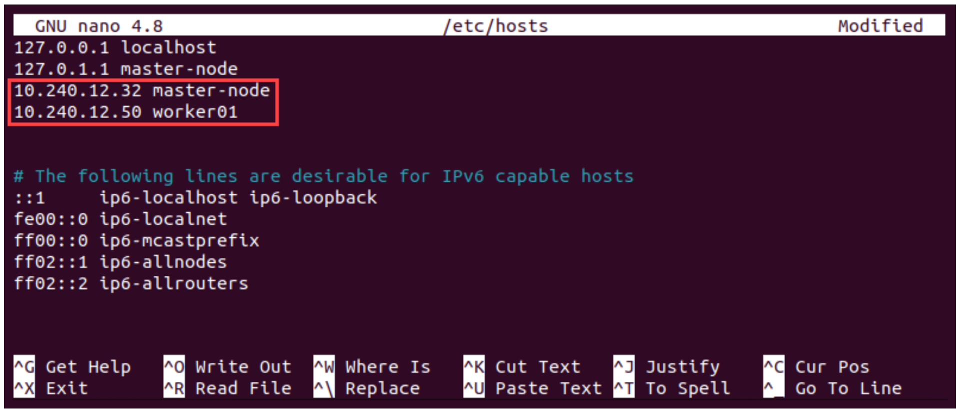
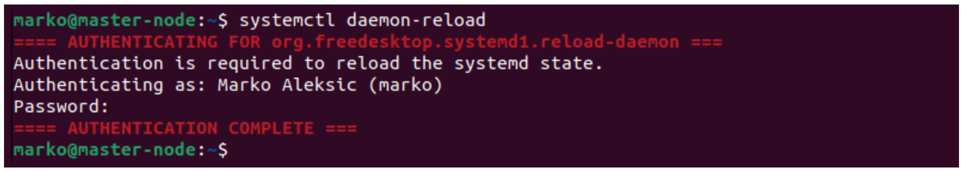
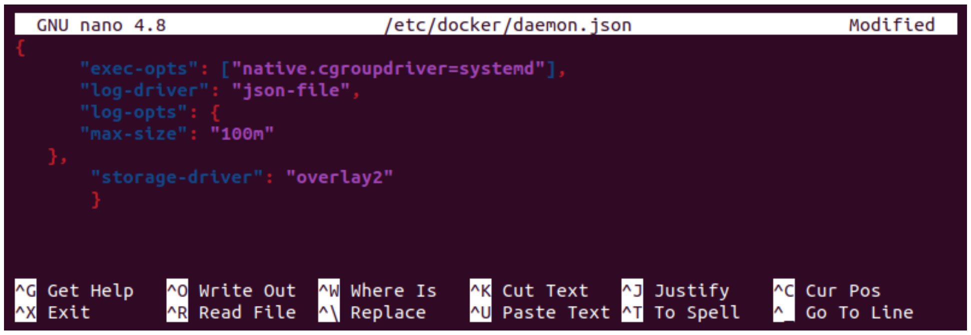
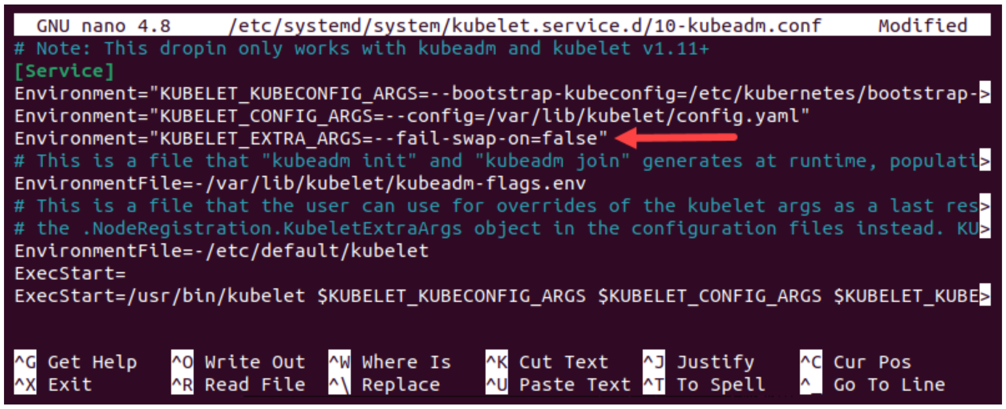
```
marko@pnap:~$ sudo apt install kubeadm kubelet kubectl -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 67 not upgraded.
Need to get 81.6 MB of archives.
After this operation, 327 MB of additional disk space will be used.
```

```
marko@pnap:~$ sudo apt-mark hold kubeadm kubelet kubectl
kubeadm set on hold.
kubelet set on hold.
kubectl set on hold.
marko@pnap:~$
```

```
marko@pnap:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.4", GitCommit:"
872a965c6c6526caa949f0c6ac028ef7aff3fb78", GitTreeState:"clean", BuildDate:"2022-11-09T
13:35:06Z", GoVersion:"go1.19.3", Compiler:"gc", Platform:"linux/amd64"}
marko@pnap:~$
```

```
  GNU nano 4.8              /etc/modules-load.d/containerd.conf              Modified
overlay
br_netfilter


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line
```

```
  GNU nano 4.8                /etc/sysctl.d/kubernetes.conf                  Modified
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line
```

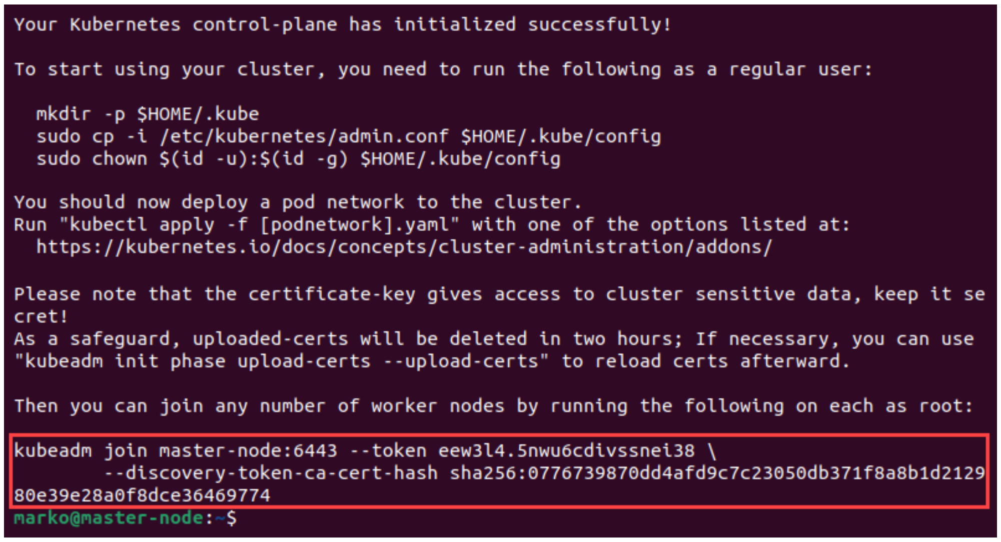
```
marko@pnap:~$ sudo sysctl --system
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
* Applying /usr/lib/sysctl.d/protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.conf ...
marko@pnap:~$
```

```
  GNU nano 4.8                       /etc/hosts                          Modified
127.0.0.1 localhost
127.0.1.1 master-node
10.240.12.32 master-node
10.240.12.50 worker01


# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

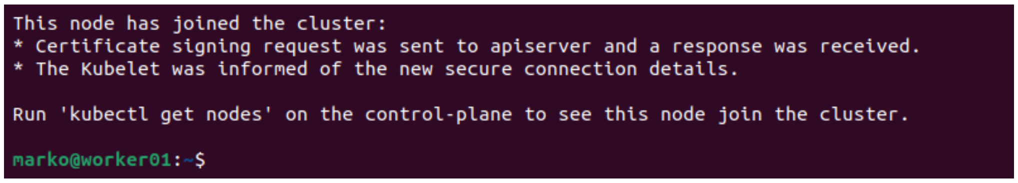
```
marko@master-node:~$ systemctl daemon-reload
==== AUTHENTICATING FOR org.freedesktop.systemd1.reload-daemon ===
Authentication is required to reload the systemd state.
Authenticating as: Marko Aleksic (marko)
Password:
==== AUTHENTICATION COMPLETE ===
marko@master-node:~$
```

```
  GNU nano 4.8                   /etc/docker/daemon.json                  Modified
{
      "exec-opts": ["native.cgroupdriver=systemd"],
      "log-driver": "json-file",
      "log-opts": {
      "max-size": "100m"
   },
       "storage-driver": "overlay2"
       }



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

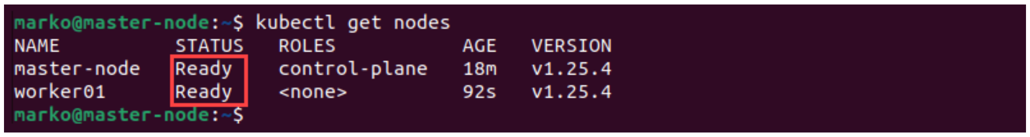
```
  GNU nano 4.8      /etc/systemd/system/kubelet.service.d/10-kubeadm.conf      Modified
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap->
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populati>
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last res>
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KU>
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBE>


^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```



```
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

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/

Please note that the certificate-key gives access to cluster sensitive data, keep it se
cret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join master-node:6443 --token eew3l4.5nwu6cdivssnei38 \
        --discovery-token-ca-cert-hash sha256:0776739870dd4afd9c7c23050db371f8a8b1d2129
80e39e28a0f8dce36469774
marko@master-node:~$
```



```
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.

marko@worker01:~$
```



```
marko@master-node:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
master-node   Ready    control-plane   18m   v1.25.4
worker01      Ready    <none>          92s   v1.25.4
marko@master-node:~$
```

Conclusion :	Kubernetes in an open source container management tool hosted by Cloud Native Computing Foundation (CNCF). This is also known as the enhanced version of Borg which was developed at Google to manage both long running processes and batch jobs, which was earlier handled by separate systems. Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container centric infrastructure.
References:	[1] https://www.tutorialspoint.com/kubernetes/index.htm [2] https://www.tutorialspoint.com/kubernetes/kubernetes_architecture.htm

Rubrics for Assessment

Timely Submission	Submitted after 2 weeks 0	Submitted after deadline 1	On time Submission 2
Understanding	Student is confused about the concept 0	Students has justifiably understood the concept 2	Students is very clear about the concepts 3
Performance	Students has not performed the Experiment 0	Student has performed with help 2	Student has independently performed the experiment 3
Development	Students struggle to run virtual machines. 0	Student can write steps the requirement stated 1	Student can write exceptional steps with his own ideas 2