

EXPERIMENT - 3

# Aim:

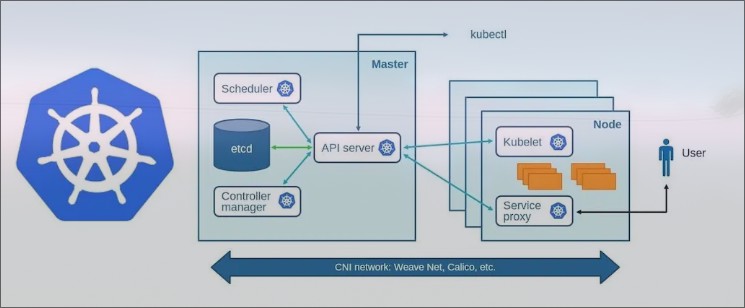
To understand the Kubernetes Cluster Architecture, install and Spin Up a

Kubernetes Cluster on Linux Machines/Cloud Platforms.

# Theory:

## Kubernetes

Kubernetes is an open source platform for managing container technologies such as Docker. Docker lets you create containers for a pre-configured image and application. Kubernetes provides the next step, allowing you to balance loads between containers and run multiple containers across multiple systems.



## Kubernetes Architecture:

Kubernetes is consists of two nodes master & worker, Nodes are the physical or virtual machines that are used to run pods. There can be only one master and multiple worker nodes.

* Master (Control plane)

Master node often reffered to as control plane and responsible for managing and orchestrating the overall operations of the system, It serves as central control point of cluster. It interact with worker node to deploy pods.

Components:

* + Schedhuler: schedule worker node for running pods.
  + Controller manager: The main function is to maintain desired state of cluster. Checks what the workers are doing and they are up.
  + API server: Directly communicated with worker from master and vice versa.
  + etcd: Store state of kubernetes cluster in key-value data store.
  + Kubectl: Kubectl is a command-line tool used to communicate with a Kubernetes cluster’s control plane using the API server. Allows to run commands to deploy

application, inspect and manage resources.

* + CNI: Container network interface.
  + Pod: This host and manage our containers that run our application.
* Worker:

Machine which runs containers and workloads. This is where the actual application is running.

Components:

* + kubelet: Stay in worker node, Manage containers and ensures they’re running as expected. It communicates with the API server to receive information about the pods that are assigned to the node.

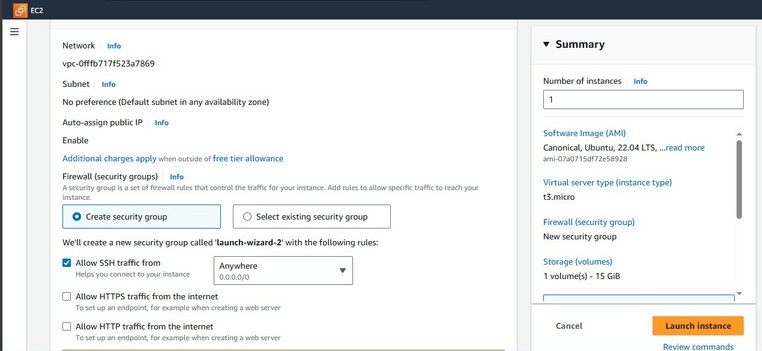
service proxy : It maintains some network rules which determines how traffic is

•

allowed to and from the Pods. It also Allow users/clients to access application.

## Steps:

1. Start by creating two Aws EC2 instance for worker & master node



1. After creating servers execute commands according on master and worker:

- Install Dependencies *# Execute On both the nodes (master and worker)*

*~$ sudo apt update*

*~$ sudo apt-get install -y apt-transport-https ca-certificates curl*

*~$ sudo apt install docker.io -y*

*~$ sudo systemctl enable --now docker*

- Install kubeadm # Execute On both the nodes (master and worker)

*~$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https:-/pkgs.k8s.io/core:/stable:/v1.28/deb/ /" | sudo tee*

*/etc/apt/sources.list.d/kubernetes.list*

*~$ curl -fsSL https:-/pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor - o /etc/apt/keyrings/kubernetes-apt-keyring.gpg*

*~$ sudo apt update*

*~$ sudo apt install kubeadm kubectl kubelet -y*

* Now, initialize kubeadm (kubernetes) in Master node.

*~$ sudo kudeadm init*

* Setup local kubeconfig # Execute on master node.

*~$ mkdir -p $HOME/.kube*

*~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config*

*~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config*

* Apply weave network # On both

[*~$ kubectl apply -f*](https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml)[*https:-/github.com/weaveworks/weave/releases/download/v2.8.1/weave-*](https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml)[*daemonset-k8s.yaml*](https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml)

* Generate token for worker node to join # On master

*~$ sudo kubeadm token create –print-join-command*

* Now, paste the token (output) of 6th step, also append --v=5 at end.

*~$ sudo kubeadm join 172.31.61.228:6443 --token f4mesu.7rzk86ga48n3uydh --discovery-token-*

*ca-cert-hash sha256:66c9863913ffdb50316e82b74f3703f73e42c4210d3a01ec7afcdbc01f677eec –v=5*



* Verify you worker node connection by running this command on Master node.

*~$ kubectl get pods*



Conclusion: Thus we successfully understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.