**Kubernetes: K8s**

**===========================================**

**It is an open source tool, which has be to be installed**

**But is also available as a service on cloud platforms as EKS, GKE, AKS, DOKS**

**It can orchestrate containers of any CONTAINER RUNTIME**

**In K8s we will first take VMs install CRI(container runtime interface) then install kubernetes**

**IN K8s we will have Master and worker node as roles**

**Orchestration commands will run on master node only.**

**Containers are scheduled only on WORKER NODES by default**

**K8s supports auto scalling - Horizintal pod autoscaler**

**K8s when on cloud supports auto scaling of cluster**

**K8s support persistent volume, persistent volume claim and external storage**

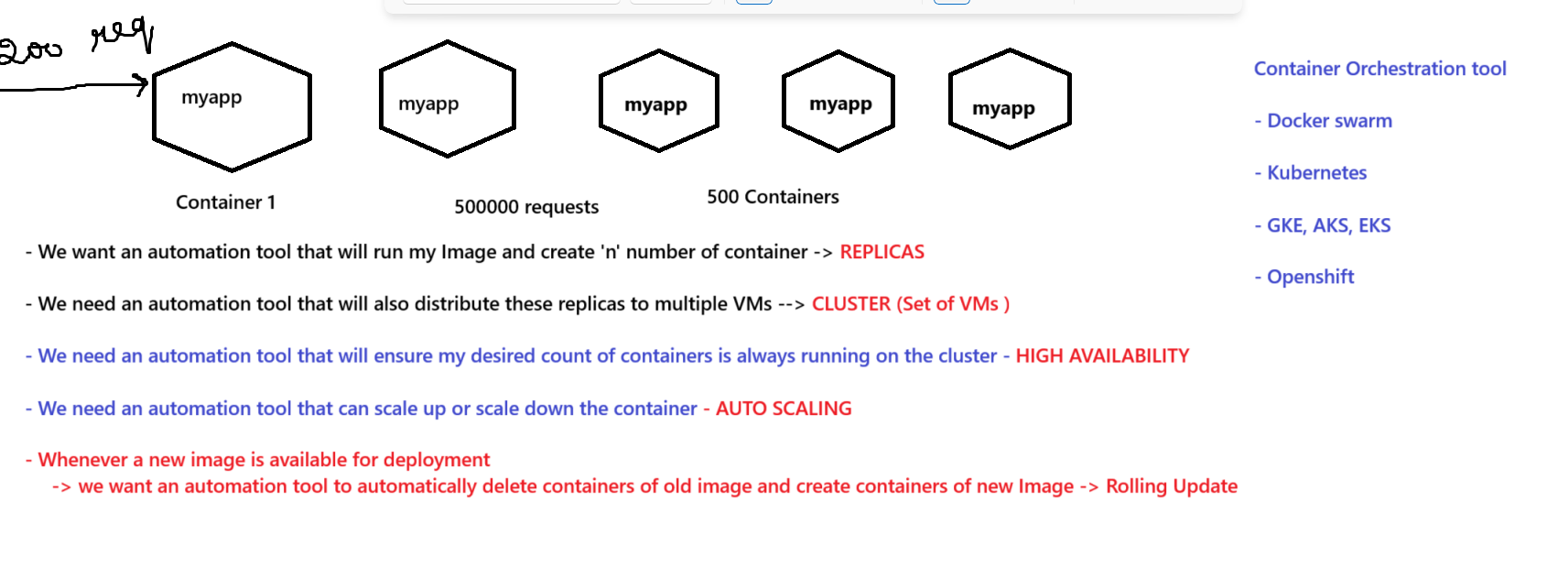
**K8s has various object and controllers to perform Orchestration activities**

**K8s supports jobs and cron jobs**

**K8s supports various scheduling techniques**

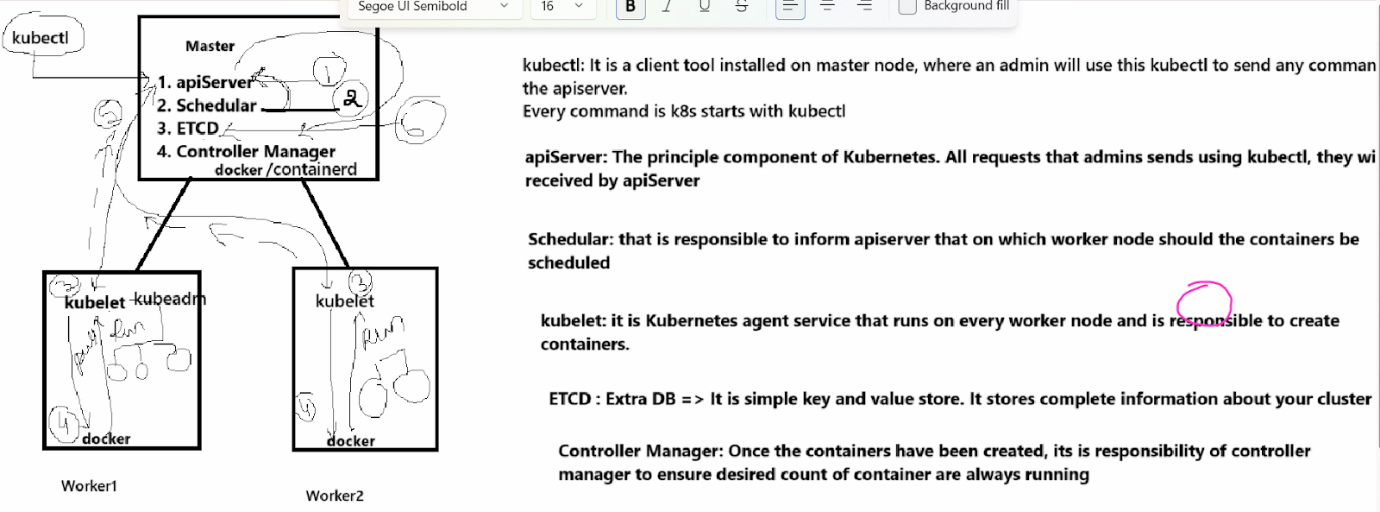
**K8s has a dashboard(GUI) to see all deployments on k8s**

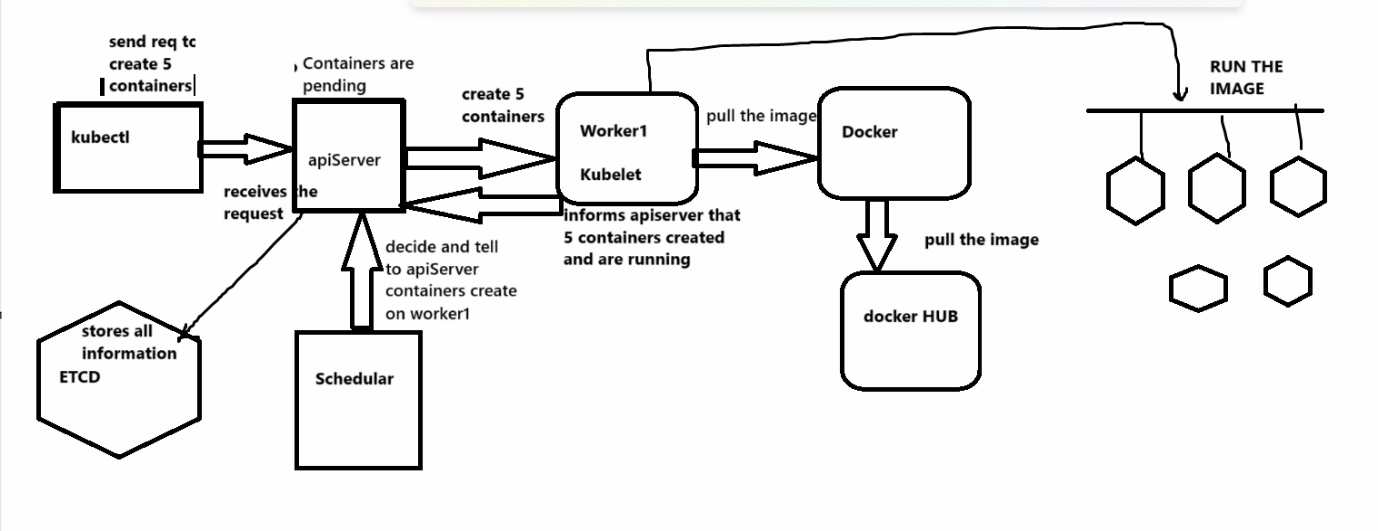
**K8s can be integrated with tools like argo cd, flux cd to manage many clusters and deploy applications in realtime**



**Kubernetes Architecture:**

**=======================**

****

****

**Install K8s in 3 ways :**

**=====================================**

**1. hard way : No longer used. we will take VM - install OS- install every component of kubernetes manually on master and worker node.**

**2. using kubeadm: It is a manual process**

**you will take the VM with Ubuntu OS -> install kubectl, kubeadm, kubelet**

**We will initiate Kubeadm --> kubeadm its a small tool that will automatically set up kubernetes components on master node**

**generates the join - token for worker**

**On the worker node we just need to install kubeadm and kubelet - sevrice is active**

**we will copy the token on worker node**

**kubeadm wills etup kubelet service on worker nodes**

**VM : 4GB and 2 CPU core on all machines**

**3. Readymade cluster**

**which are available on the cloud as a kubernetes service - EKS (AWS), GKE(GCP), AKS (azure)**

**you just need to go to the cloud platform-->provide the cluster details and click onc reate button**

**automatically the cluster will ready. this cluster is maintained by cloud**

**Auto scale of cluster can also be set**

**==============================================**

**Minimum requirement for install kubernetes is:**

* **Preferable OS - Ubuntu 22**
* **Configuration: 4GB RAM and 2 CPU core**

Kubernetes can be installed on a VM and can also be used as a service on various cloud providers:

**-> EKS on AWS**

**-> GKE on GCP**

**-> AKS on azure**

These are readymade clusters provided by cloud as a service (you pay for the kubernetes service)

As part of this service we will have a cluster with 1 master node service and user provider worker nodes.

In readymade cluster -> you will get CRI, kubernetes, CNI

But if we take our own VMs then we have to:

Install docker and container-d

Install kubernetes component

Initiate kubernetes

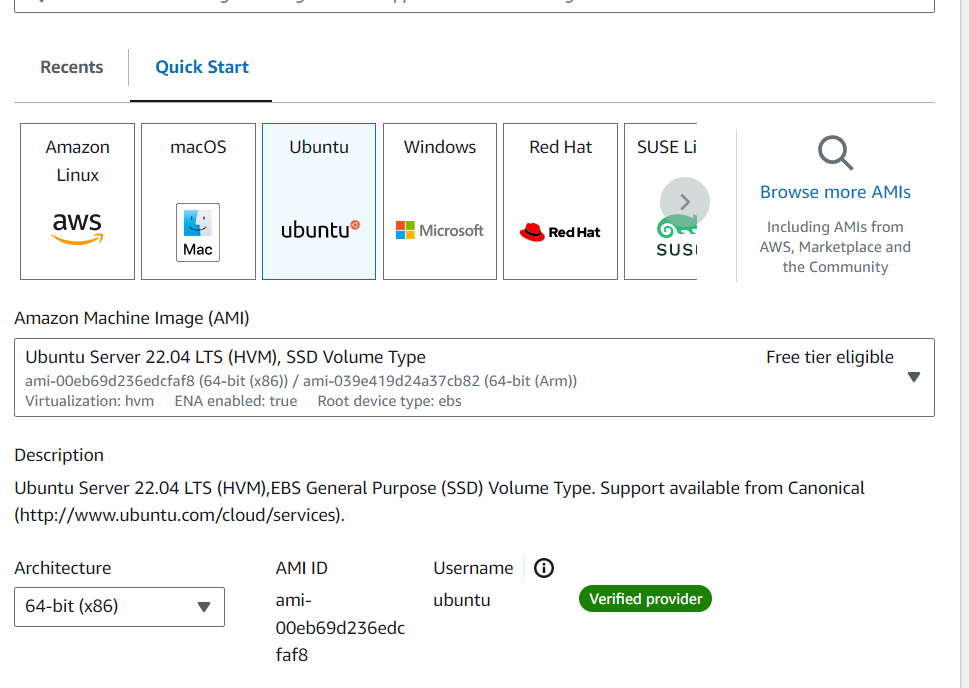
Install Container network interface

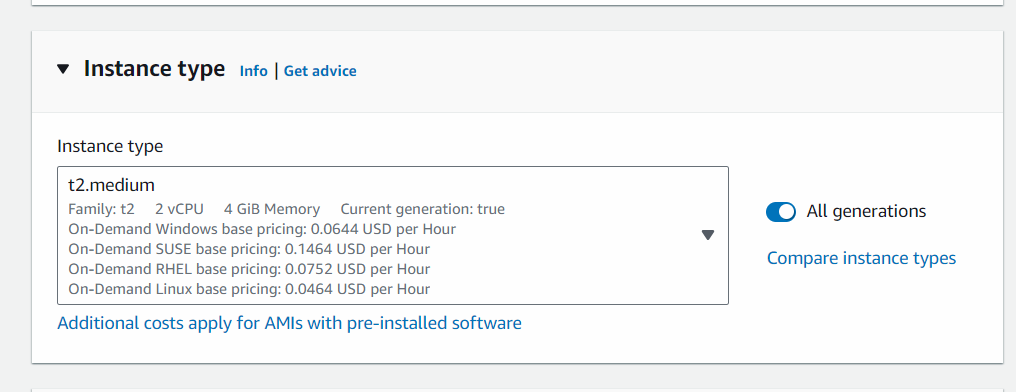
We will use a tool called kubeadm to automatically set up kubernetes components for us.

SetUp kubernetes Master Node:

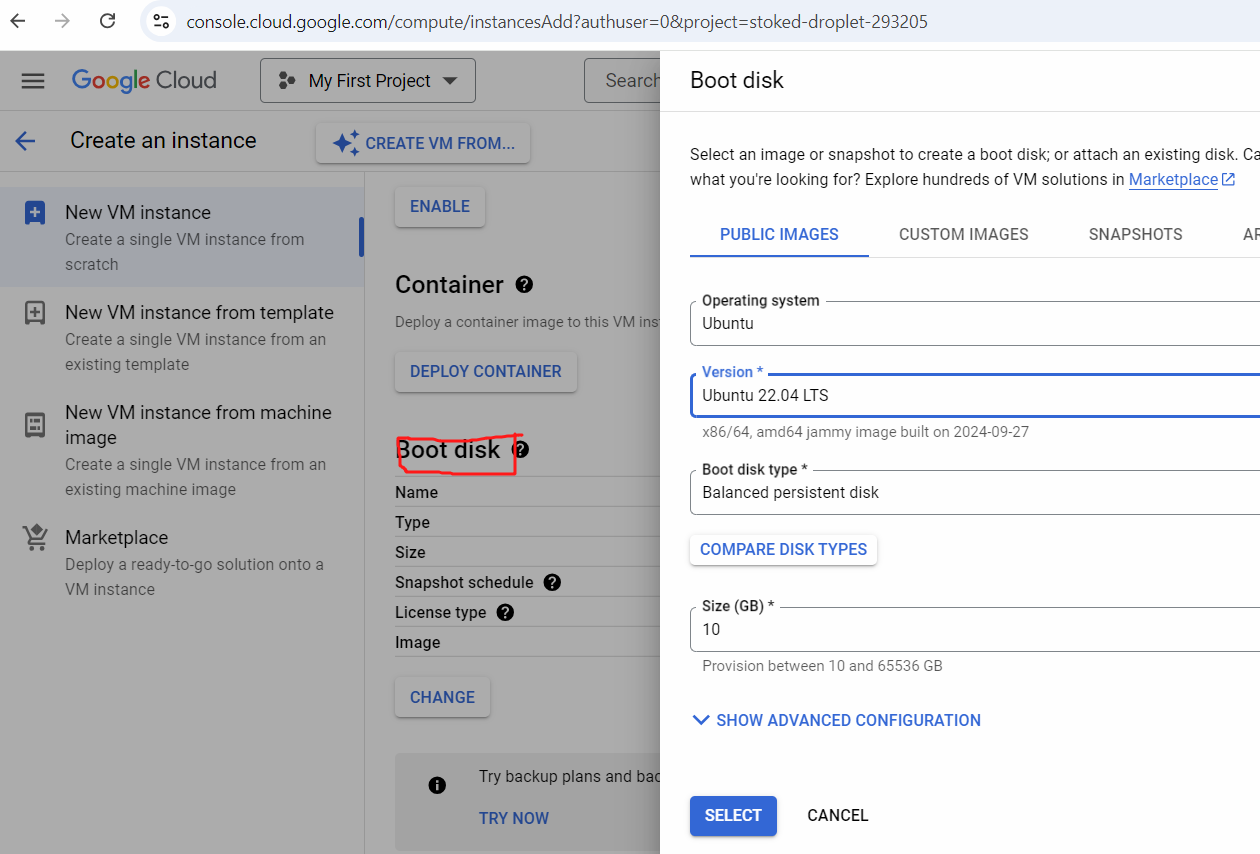
======================================

Use AWS or GCP and we will create Ubuntu servers of 4 GB RAM 2 CPU core → t2.medium





In GCP



Connect to master and execute below commands:

**Only on MASTER NODE:**

============================================

# sudo su -

**## Install Containerd**

**sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installContainerd.sh -P /tmp**

**sudo bash /tmp/installContainerd.sh**

**sudo systemctl restart containerd.service**

**### Install kubeadm,kubelet,kubectl**

You will install these packages on all of your machines:

* kubeadm: the command to bootstrap the cluster.
* kubelet: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
* kubectl: the command line util to talk to your cluster.

**sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installK8S.sh -P /tmp**

**sudo bash /tmp/installK8S.sh**

**## Initialize kubernetes Master Node**

   # **sudo kubeadm init --ignore-preflight-errors=all**

**Execute the below commands to setup kubectl and apiserver communication**

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

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

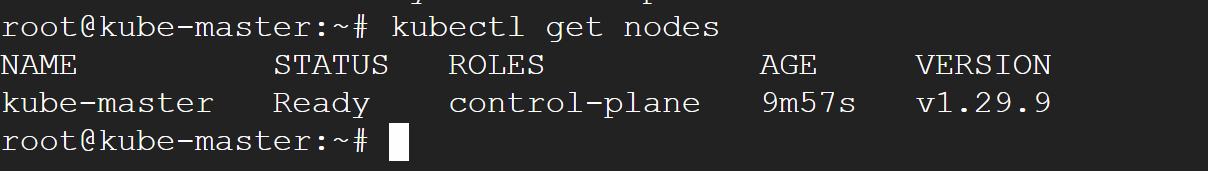
**sudo chown $(id -u):$(id -g) $HOME/.kube/config**

   ## install networking driver -- Weave/flannel/canal/calico etc...

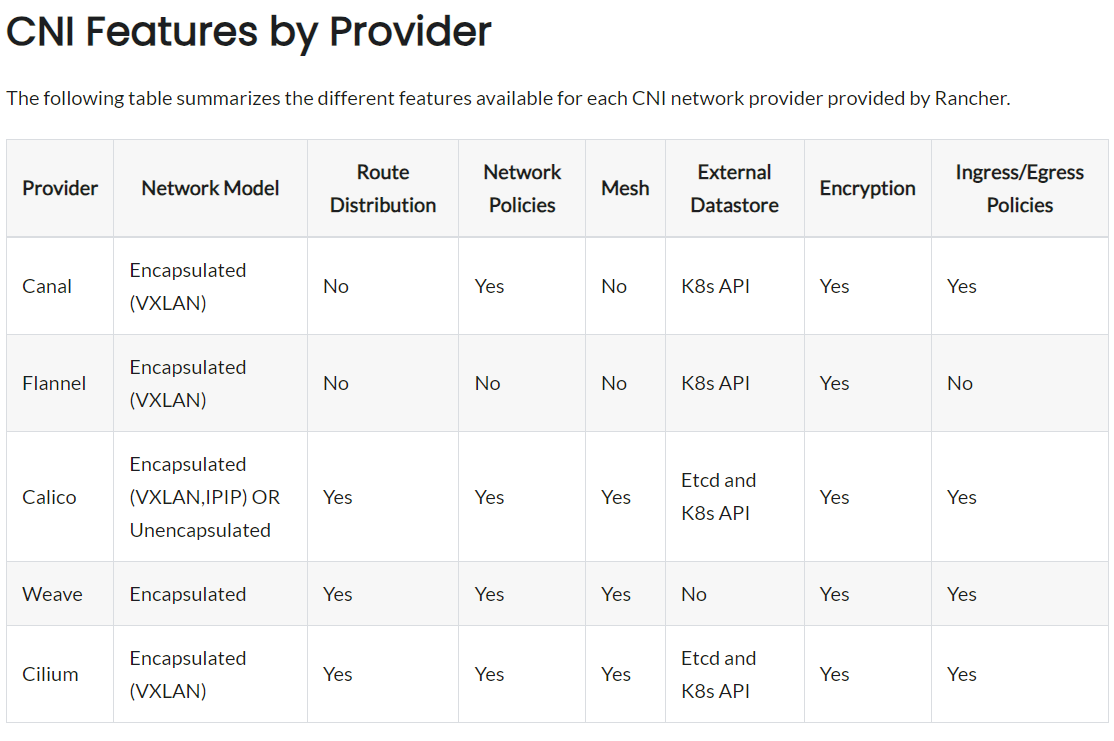
   ## below installs calico networking driver

   kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.24.1/manifests/calico.yaml

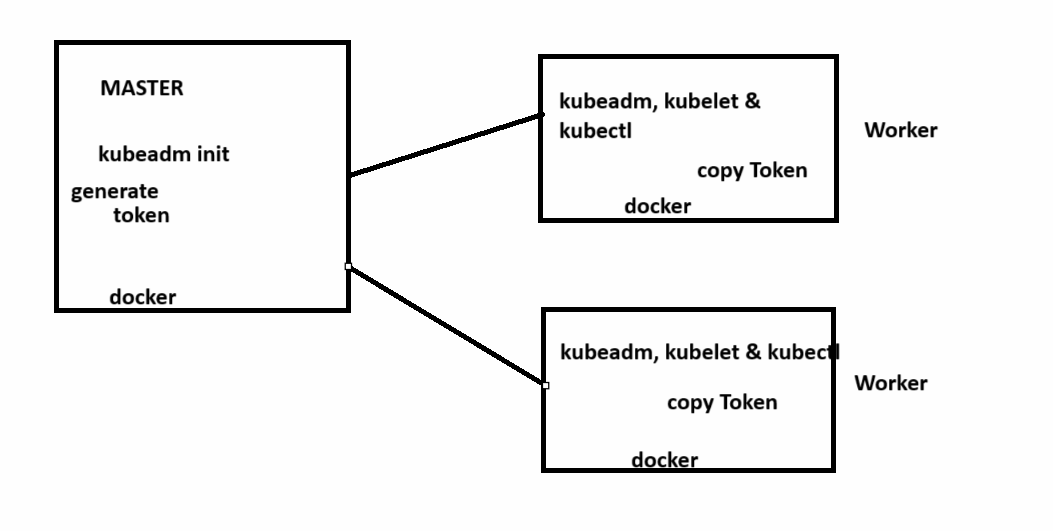
   # Validate:  kubectl get nodes



Just for your information:



=========================================

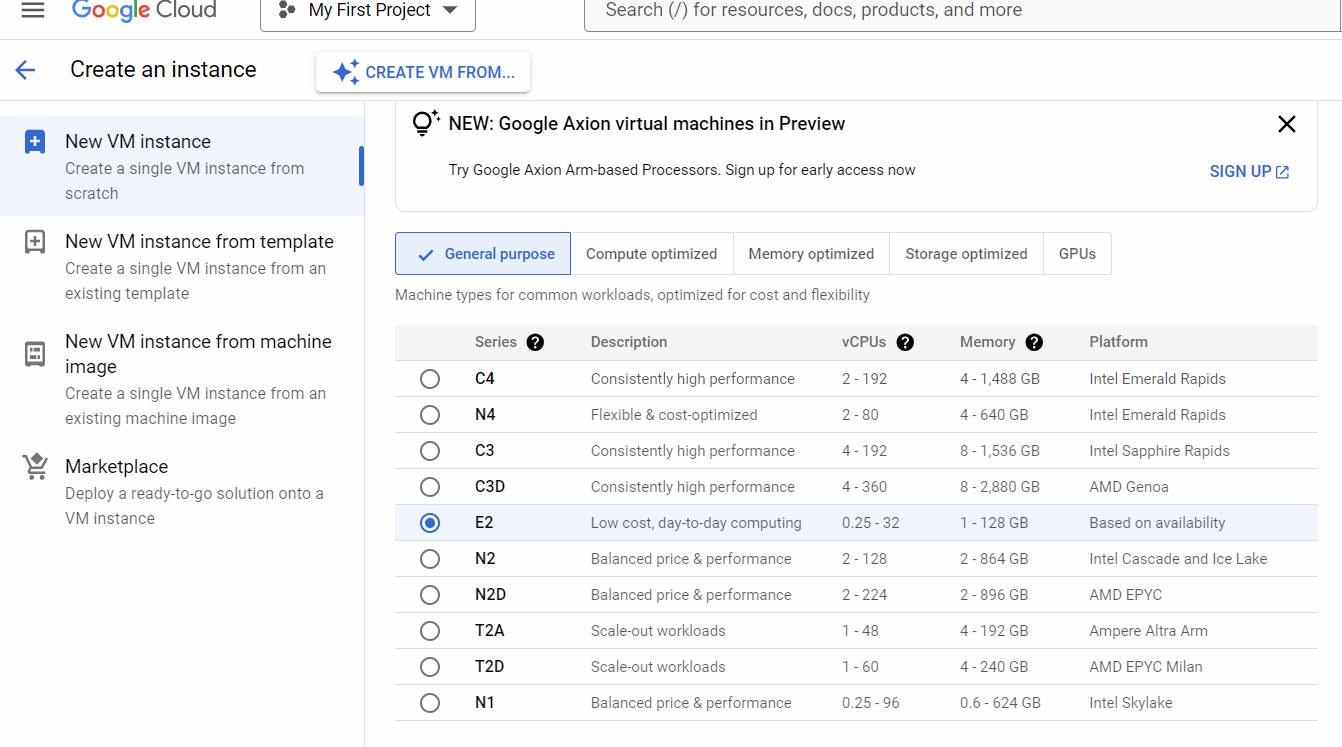


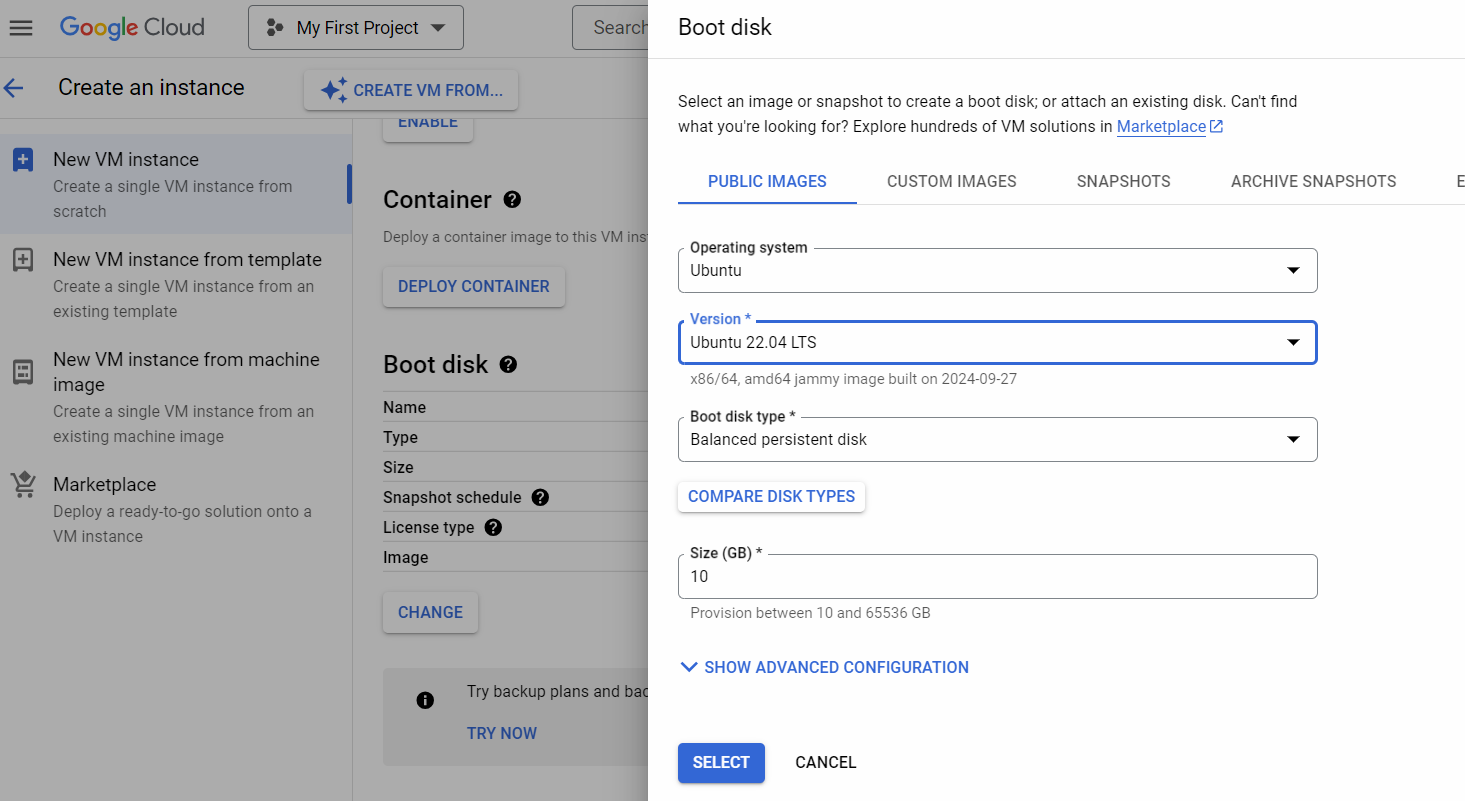
SETUP worker Nodes:

=====================================

Create an Ubuntu server:

* instance type t2.medium(AWS), e2 medium(GCP)
* OS : ubuntu 22 (AWS & GCP)





**On All Worker Nodes**

**## Install Containerd**

**# sudo su -**

sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installContainerd.sh -P /tmp

sudo bash /tmp/installContainerd.sh

sudo systemctl restart containerd.service

**## Install kubeadm,kubelet,kubectl**

sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installK8S.sh -P /tmp

sudo bash /tmp/installK8S.sh

**## Run Below on Master Node to get join token**

# **kubeadm token create --print-join-command**

    copy the kubeadm join token from master & run it on all worker nodes

On Master node:

# kubectl get nodes

==================================

Kubernetes APIVERSION document:

<https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.32/#-strong-api-groups-strong->

**In k8s the pods will always be scheduled on the worker node**

**In kubernetes we write the code to create Pod using a single line command or using a manifest file.**

**# kubectl run pod1 --image nginx**

**# kubectl get pods**

**# kubectl get pods -o wide**

**# kubectl describe pod pod1 | less**

**# kubectl delete pod pod1**

**Press q to come out of the above command.**

**===========================================**

**Create a Objects using a manifest YAML file.**

**=========================================**

[**https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.31/#-strong-api-groups-strong-**](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.31/#-strong-api-groups-strong-)

**FIELDS:**

**apiVersion    <string>**

**APIVersion defines the versioned schema of this representation of an object.**

**Servers should convert recognized schemas to the latest internal value, and**

**may reject unrecognized values. More info:**

[**https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources**](https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources)

**kind  <string>**

**Kind is a string value representing the REST resource this object**

**represents. Servers may infer this from the endpoint the client submits**

**requests to. Cannot be updated. In CamelCase. More info:**

[**https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds**](https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds)

**metadata      <ObjectMeta>**

**If the Labels of a ReplicaSet are empty, they are defaulted to be the same**

**as the Pod(s) that the ReplicaSet manages. Standard object's metadata. More**

**info:**

[**https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata**](https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata)

**spec  <ReplicaSetSpec>**

**Spec defines the specification of the desired behavior of the ReplicaSet.**

**More info:**

[**https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status**](https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status)

**Demo1:**

**====================**

**# mkdir mykubefiles**

**# cd mykubefiles**

**# vim pod-defintion.yml**

**kind: Pod**

**apiVersion: v1**

**metadata:**

**name: pod2**

**labels: # any key and value pair**

**author: sonal**

**type: webserver**

**env: dev**

**spec:**

**containers:**

**- name: c1**

**image: nginx**

**- name: c2**

**image: tomcat**

**- name: c3**

**image: ubuntu**

**command: ["bash", "-c", "sleep 6000"]**

**# kubectl create -f pod-defintion.yml**

**# kubectl get pods**

**# kubectl logs pod2 -c c1**

**Get number of containers in a pod:**

**# kubectl get pods pod2 -o jsonpath='{.spec.containers[\*].name}'**

**# kubectl delete pod pod2**