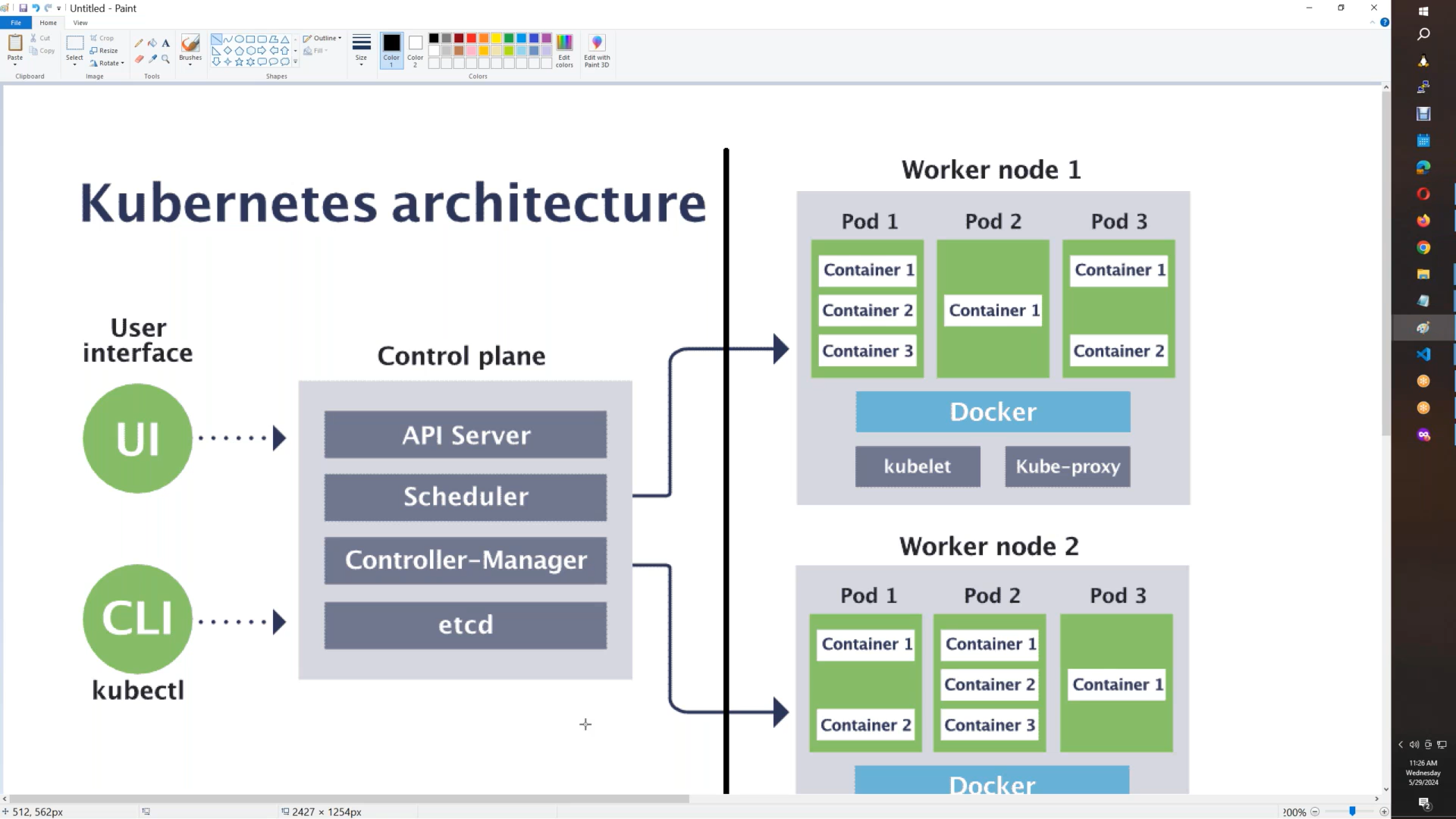
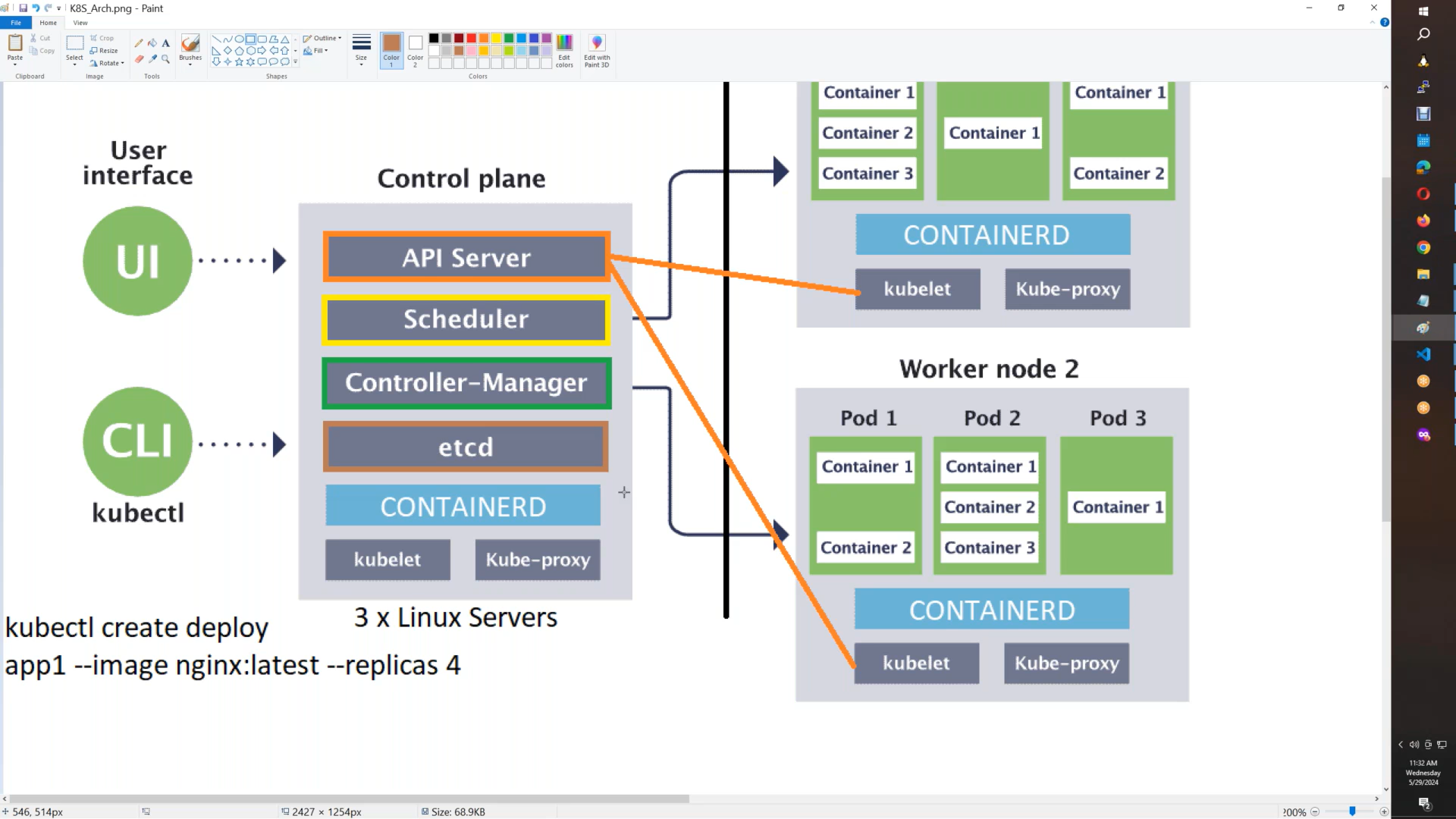
**Kubernetes**

**Architecture**





**control plain**

should install in linux only

**Worker nodes**

Which helps us to deploy our applications

**Components of control plain**

**API Server** --> which is responsible for One-stop connectivity for both Administration & kubelet

**Scheduler** 🡪 for scheduling the pods

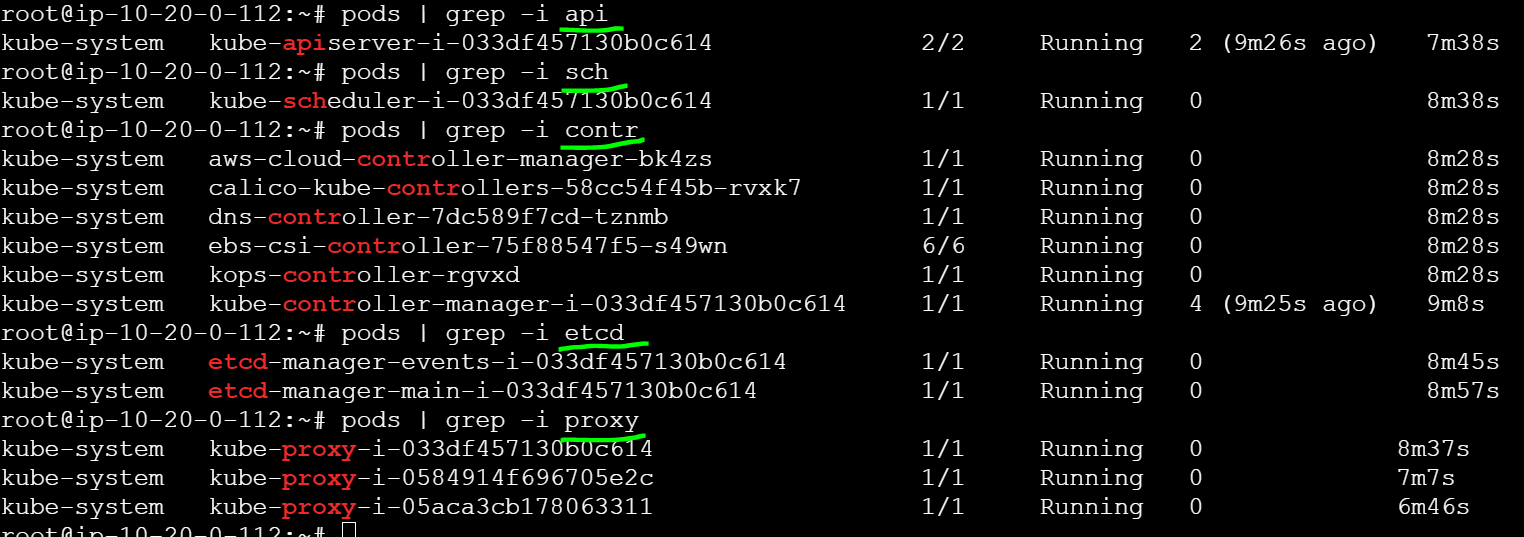
**Controller-manager 🡪** to control replicas, end-points, nodes

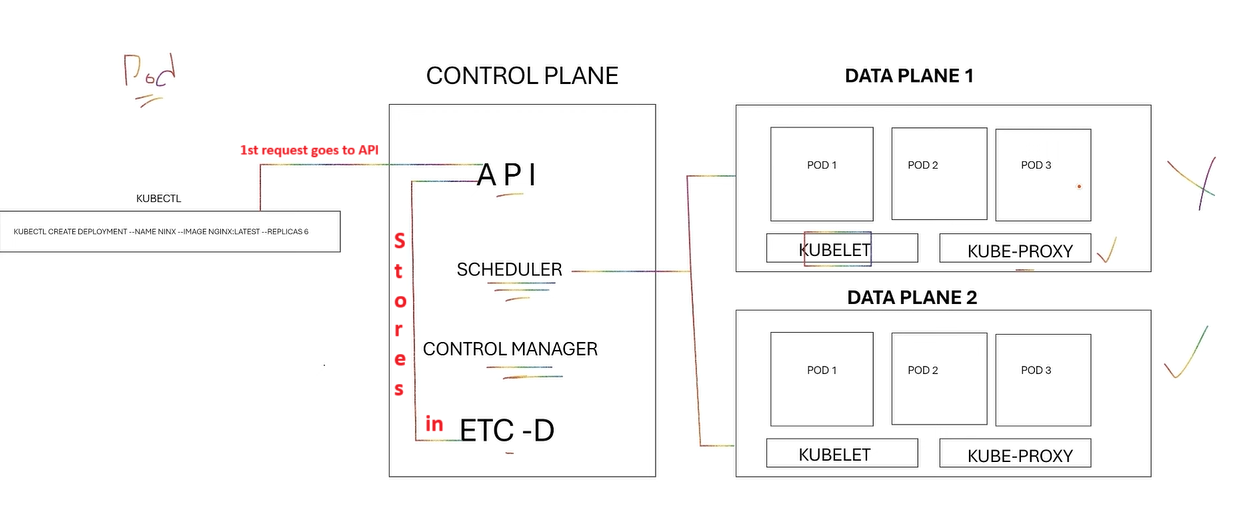
**etcd 🡪** a distributed database which has only connectivity from API server & it is encrypted.

**Kube-proxy 🡪** Responsible for ip tables ( firewall manipulate / traffic forward /

**Internally** API Server/ Scheduler/controller-manager/etcd/kube-proxy runs as **Container (pods)**

Kubelet runs as a **Service (like ssh , http )**



****

**Kubectl** is a command line tool which is used to run commands against your Kubernetes(K8s) cluster.

**Simple language – you are going to communicate K8s cluster using Kubectl**

**API :**

It always interacts with the external world

It’s also responsible for **server-handling request**

It’s also responsible for **state management**

It’s also responsible for **Authentication & Authorization**

**Whenever request comes from Kubectl Api will validate it & stores this information in ETC-D**

**Scheduler:** is responsible for scheduling the pods in multiple worker nodes & also it will have all the information of worker node1 & worker node2

**Control-Manager: (node controller & Replication controller)**

**Incase if the worker node1 is down under the control manager there will be a node controller & Replication controller they will work together & re-schedule all the pods in the failed worker node to Active worker node.**

**Kube-proxy 🡪** Responsible for ip tables ( firewall manipulate / traffic forward /

(kube-proxy will assign a unique ip address to each pod)

**Internally** API Server/ Scheduler/controller-manager/etcd/kube-proxy runs as **Container (pods)**

Kubelet runs as a **Daemon** **Service (like ssh , http ) – for some reason, pods are stopped then kublet will re-start it & report to control plain**

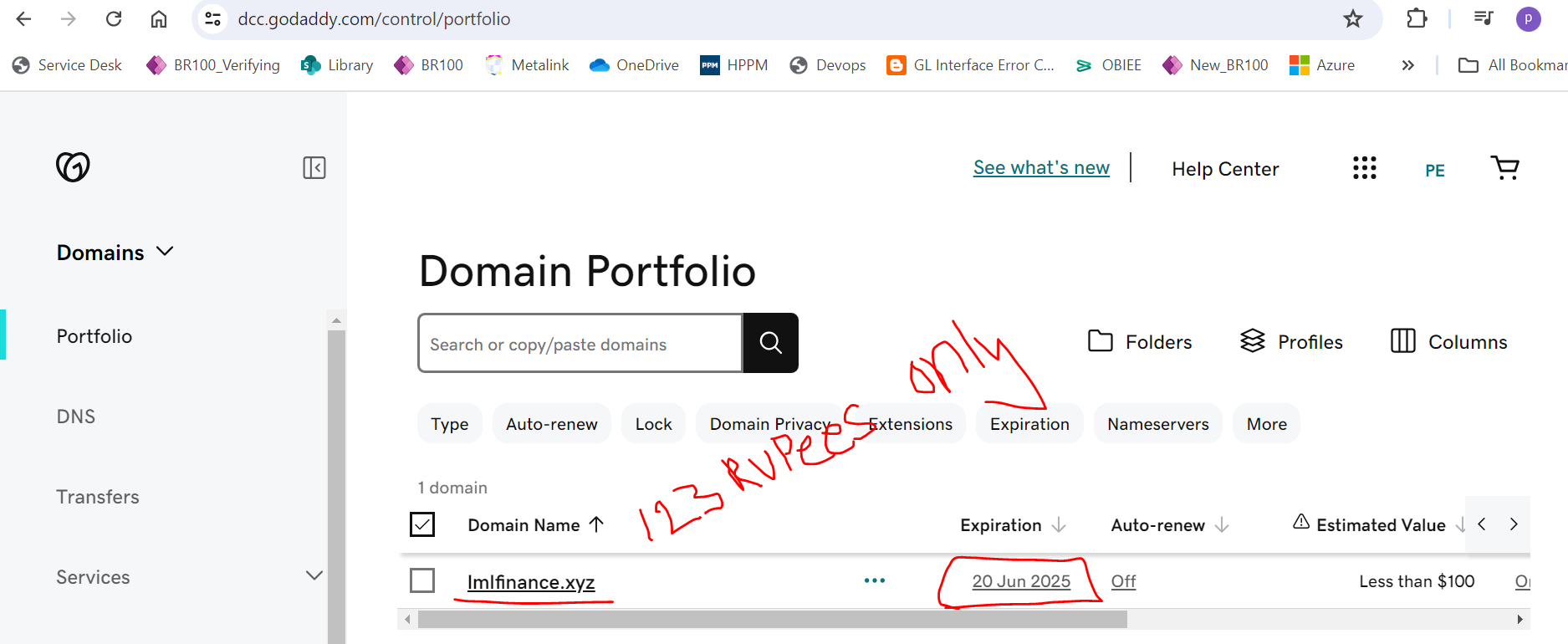
**Kops – Installation (Kubernetes Operations)**

In-Backend if you deploy worker/master node via Kops – as per the Kops functionality it will be setup auto-scaling group. advantage ( if someone tries to delete the Master/worker node auto-scaling group will automatically it will re-create the instances)

<https://github.com/kubernetes/kops/releases>

wget https://github.com/kubernetes/kops/releases/download/v1.29.0/kops-linux-amd64

Purchased Domain name for 1 year from GoDaddy Expiry date



Go to Route53 & copy the name servers & paste it to Godady domain

Create VPC

1. Availability Zone 1
2. Public subnet 1
3. Private subnet 0
4. Update CIDR block
5. NAT gateway none
6. VPC end-points none

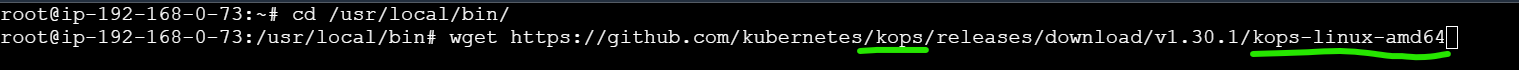
Create IAM role – give full admin rights

Create EC2 with above VPC & assign IAM role

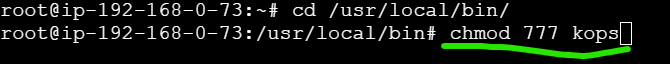
Login to ec2 install & generate ssh-keygen

Install tools – apt update && apt install -y jq net-tools unzip

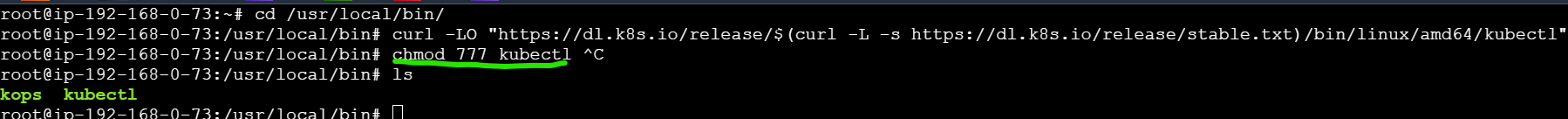
Install Kops -- <https://github.com/kubernetes/kops/releases/download/v1.30.1/kops-linux-amd64>

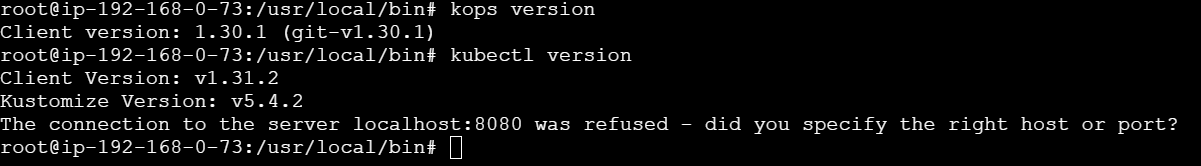






**Install Kubectl** -- curl -LO "https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl"





# open this file & paste the below details

vim .bashrc

export CLUSTER\_NAME=lmlfinance.xyz

export AWS\_REGION=ap-southeast-2

export NAME=lmlfinance.xyz

export KOPS\_STATE\_STORE=s3://lmlfinance.xyz

export KUBE\_EDITOR=vim

# to create cluster

kops create cluster --name=lmlfinance.xyz \

  --state=s3://lmlfinance.xyz \

  --zones=ap-southeast-2a,ap-southeast-2b \

  --node-count=1 \

  --control-plane-count=3 \

  --control-plane-size=t3.medium \

  --control-plane-zones=ap-southeast-2a,ap-southeast-2b \

  --control-plane-volume-size 10 \

  --node-volume-size 10 \

  --node-size=t3.medium \

  --ssh-public-key ~/.ssh/id\_rsa.pub \

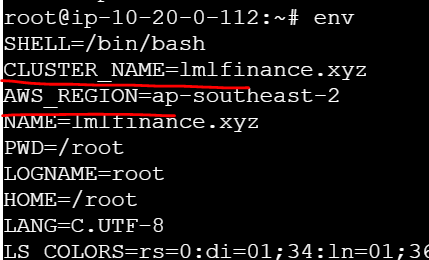
  --dns-zone=lmlfinance.xyz \

  --networking calico  \

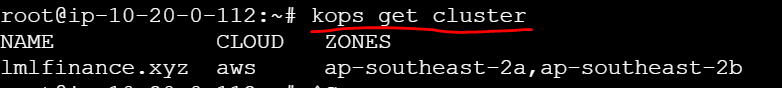
  --dry-run -o yaml > lmlfinance.yaml

  # --networking calico --yes (if you are not use --yes cluster will not deploy created as )

**#validation**

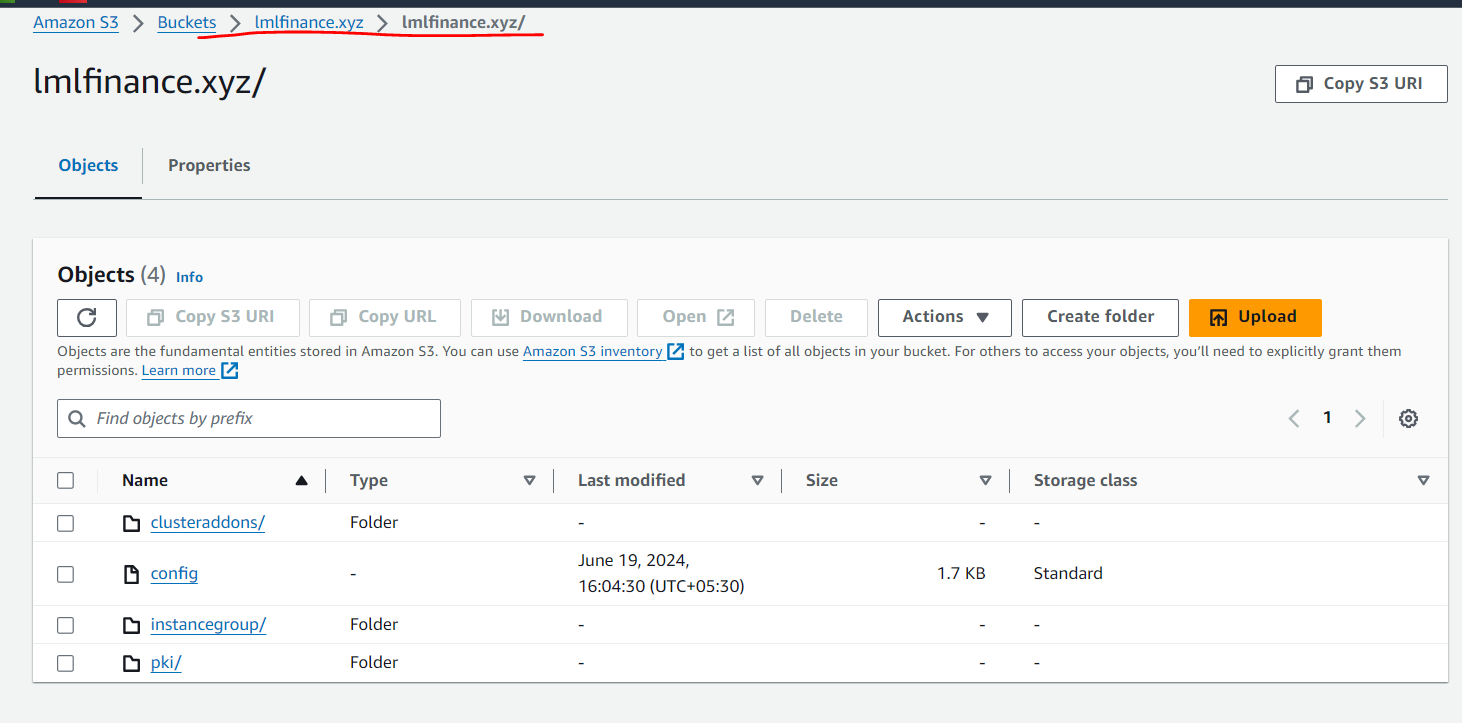
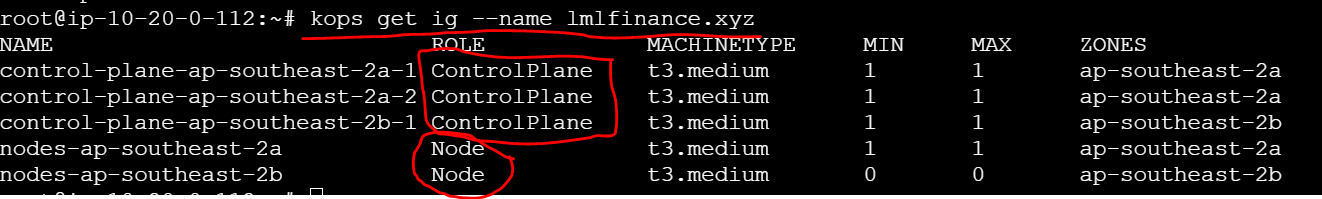


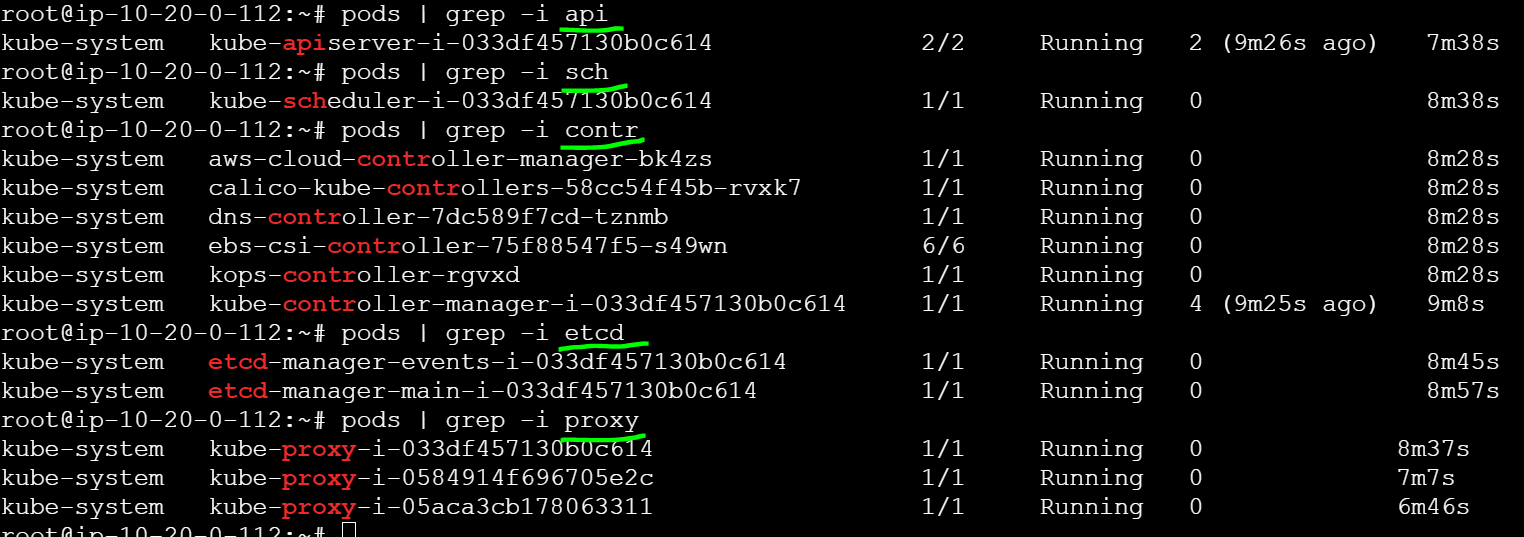
kops get cluster



kops edit cluster lmlfinance.xyz #to edit security group details/cidr block

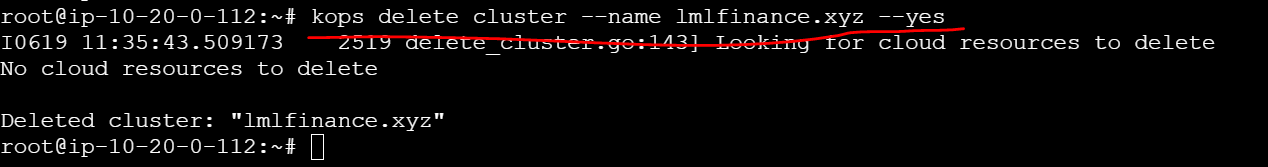


kops get ig --name lmlfinance.xyz # ig = instance group

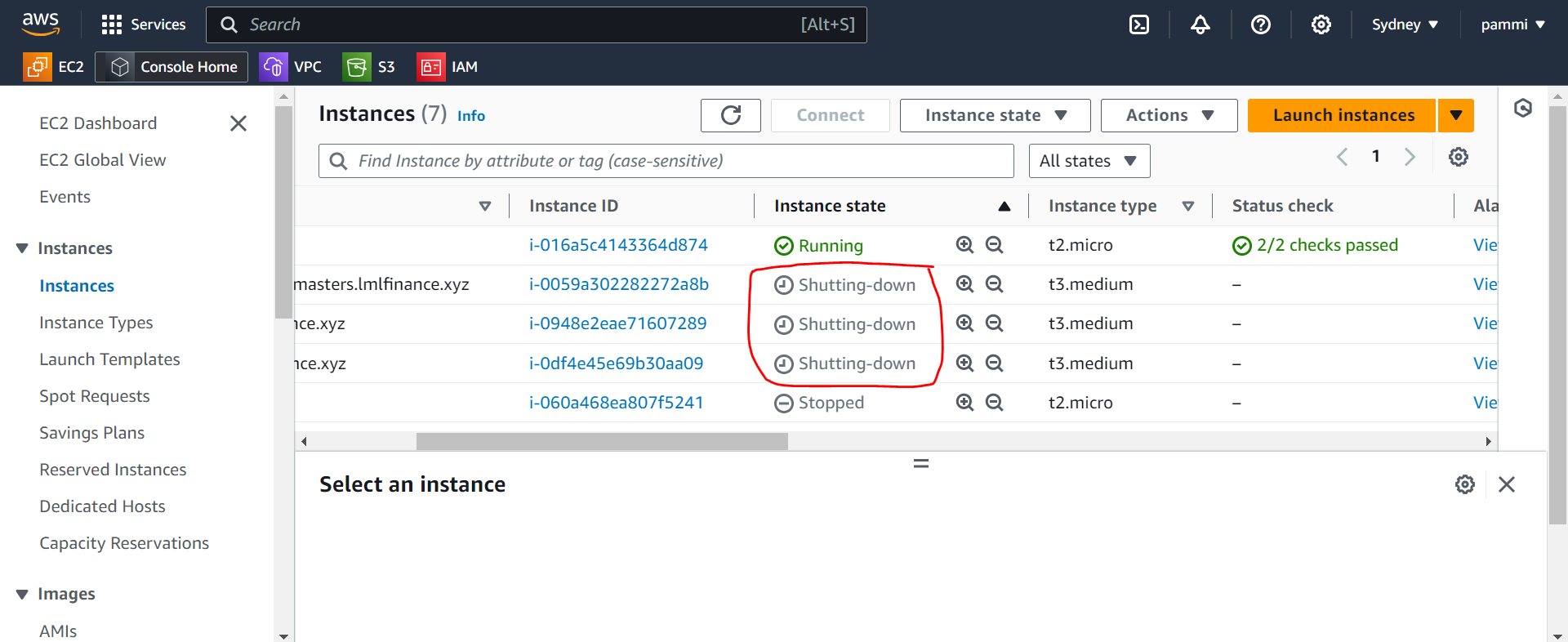


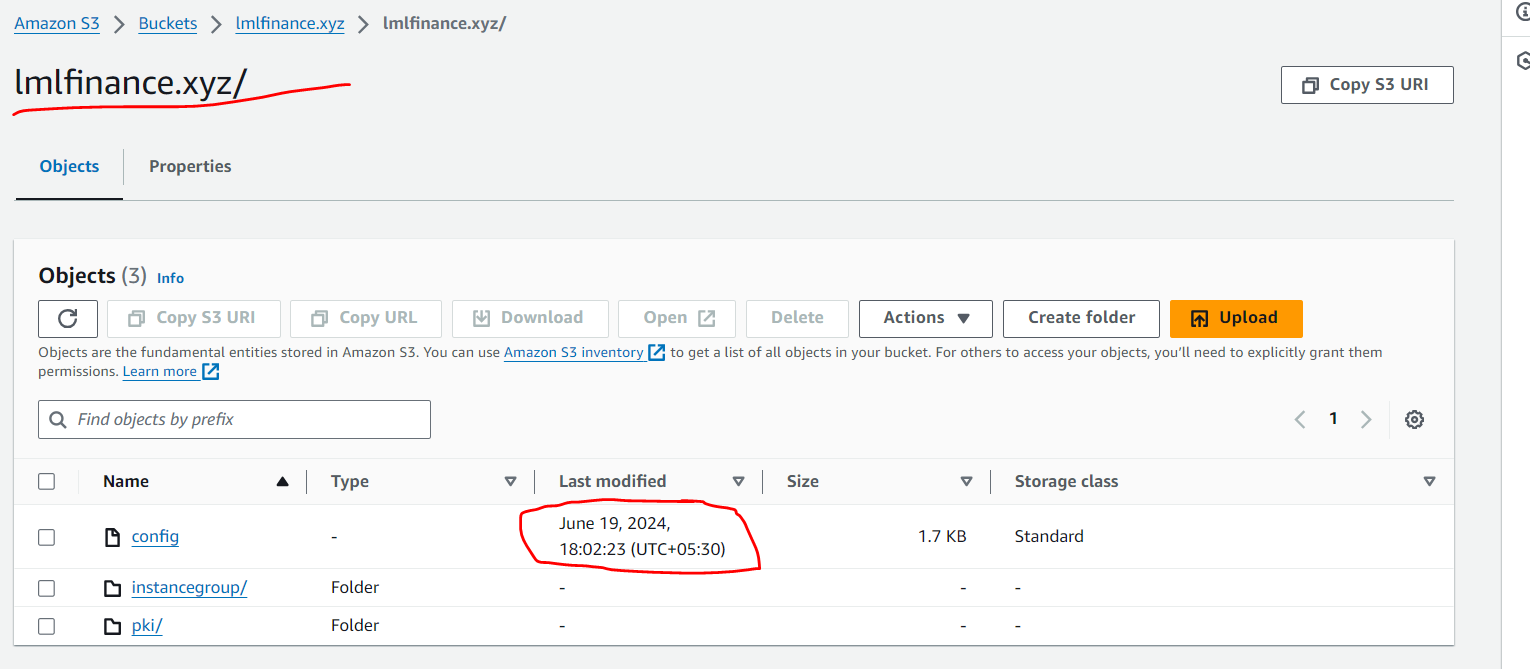
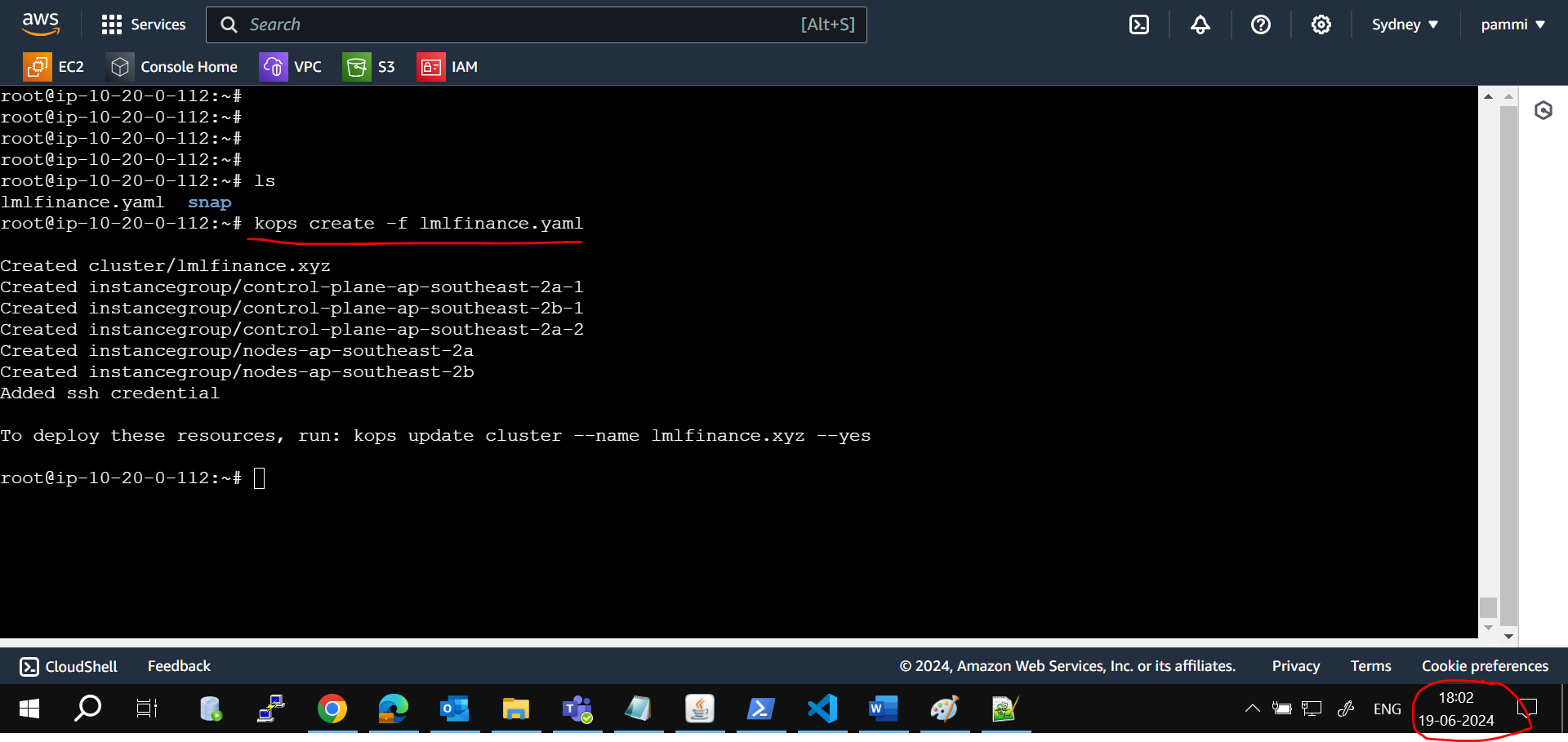
**Delete cluster**

kops delete cluster --name lmlfinance.xyz --yes

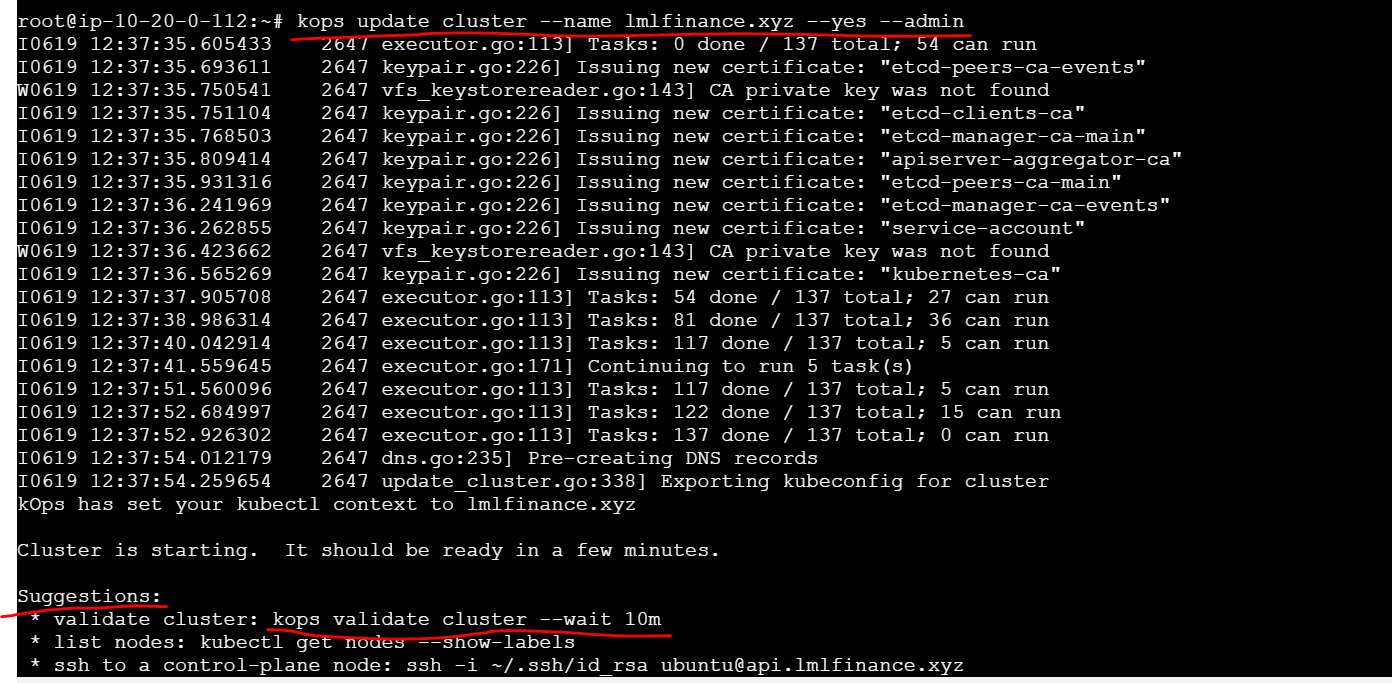


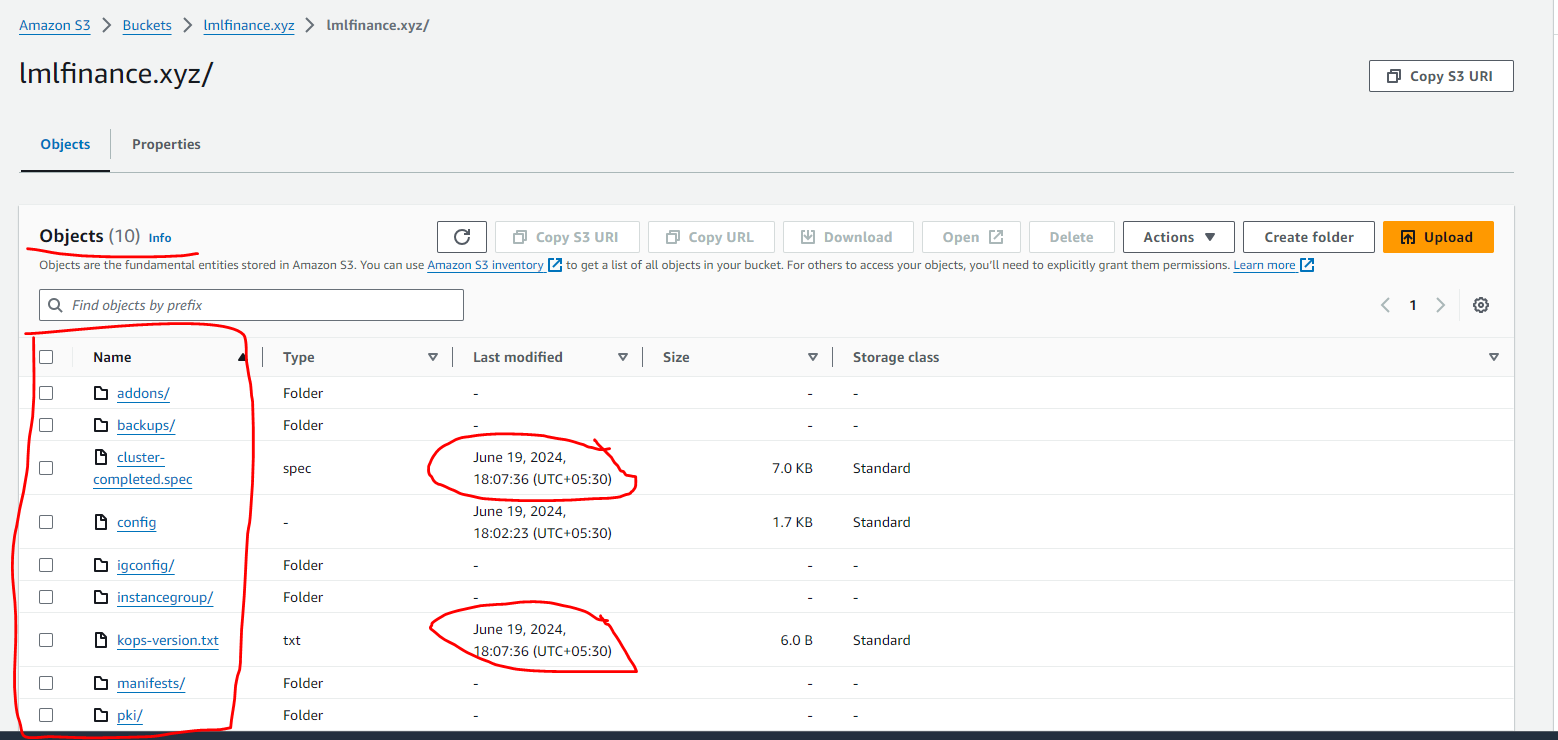




kops create -f lmlfinance.yaml

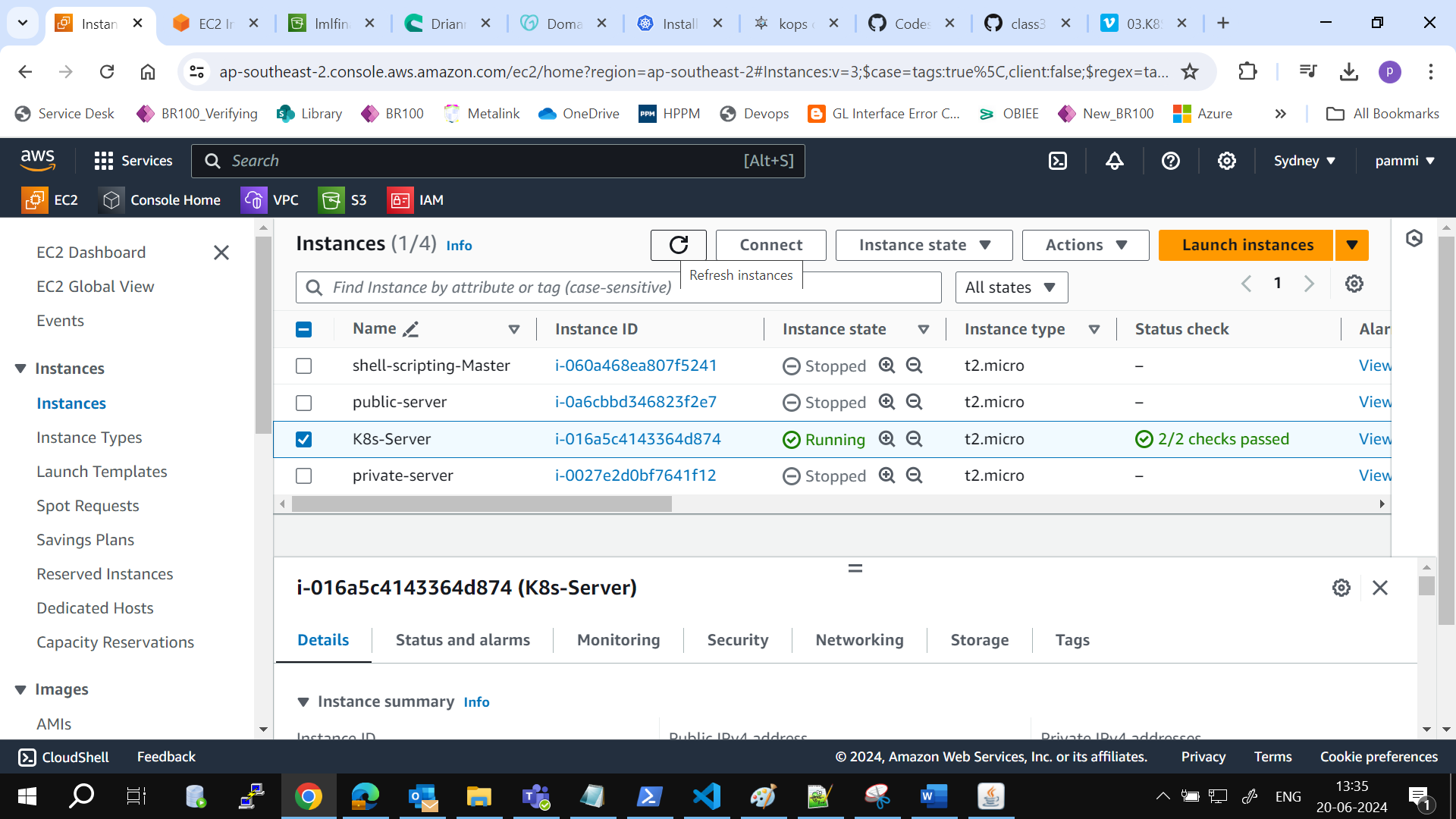
kops update cluster --name lmlfinance.xyz --yes –admin





**Class3**

**Before**





**After**

#deploy cluster with 1 control plane & 2 worker nodes

#first while practice please apply this below command for clean details of .kube location

# rm -rf .kube/config

#1 master (control-plane-count) -2 worker nodes (node-count) cluster

kops create cluster --name=lmlfinance.xyz \

  --state=s3://lmlfinance.xyz \

  --zones=ap-southeast-2a,ap-southeast-2b \

  --node-count=2 \

  --control-plane-count=1 \

  --control-plane-size=t3.medium \

  --control-plane-zones=ap-southeast-2a \

  --control-plane-volume-size 10 \

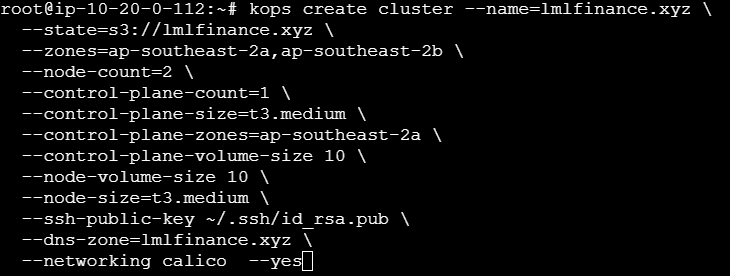
  --node-volume-size 10 \

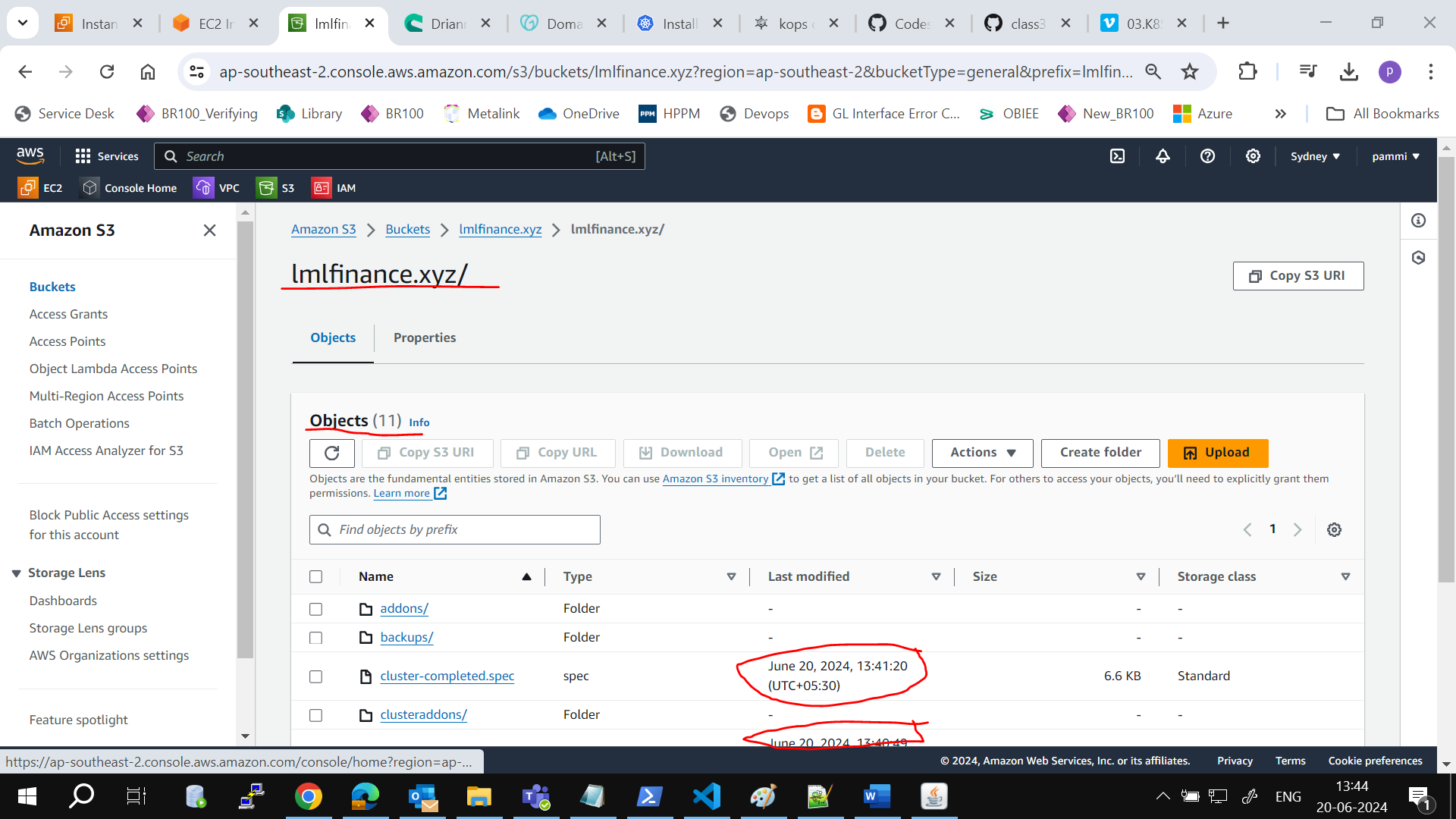
  --node-size=t3.medium \

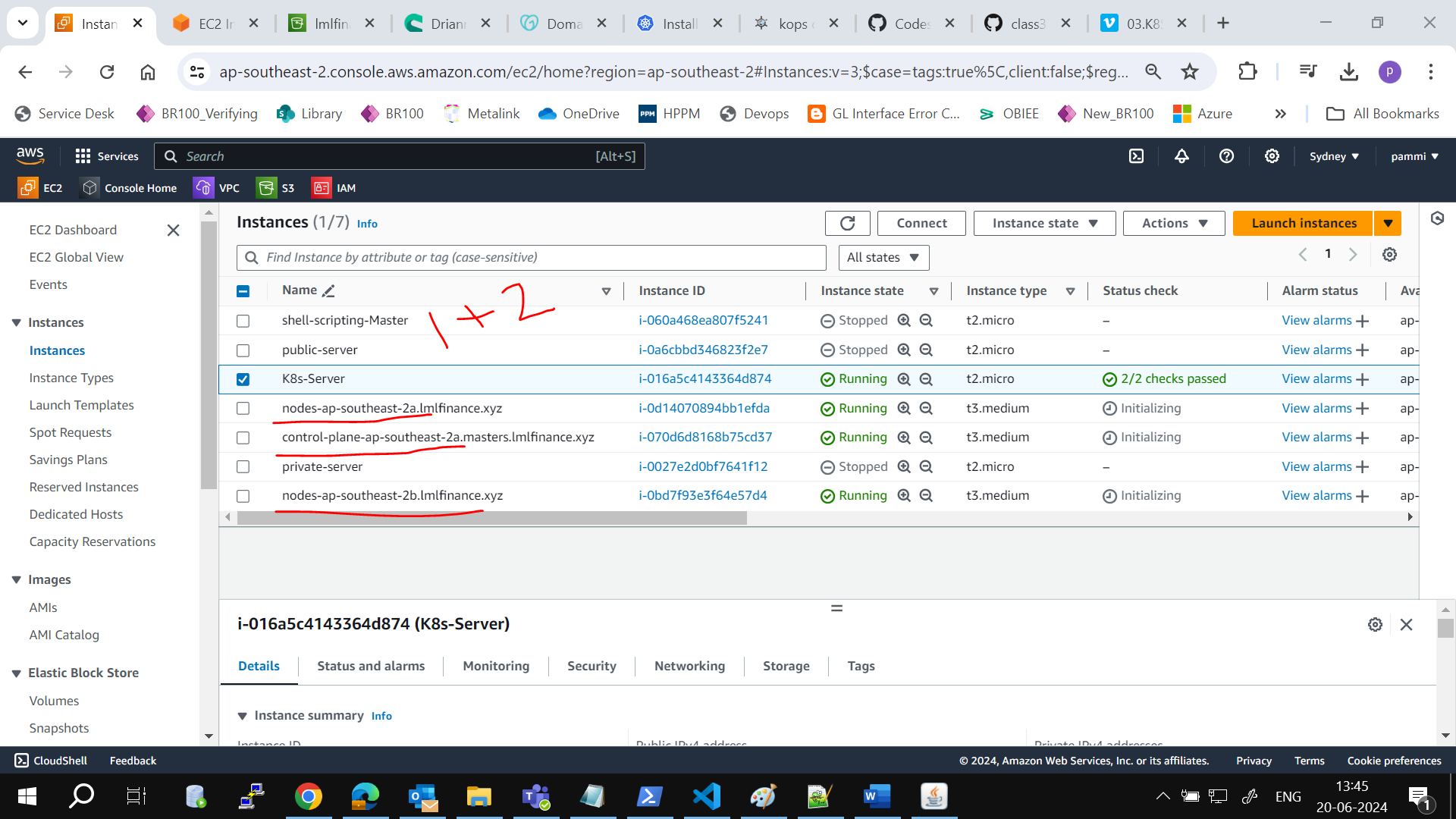
  --ssh-public-key ~/.ssh/id\_rsa.pub \

  --dns-zone=lmlfinance.xyz \

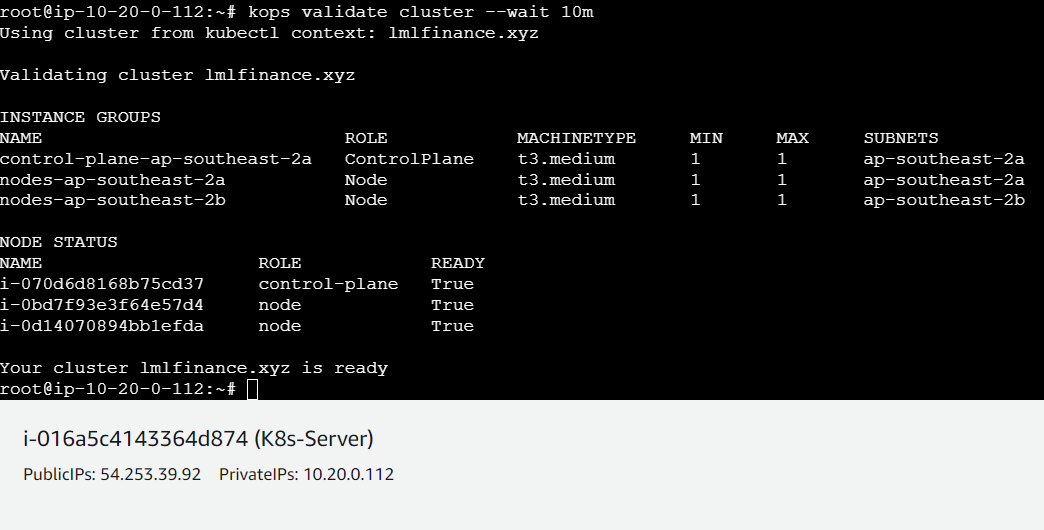
  --networking calico  --yes

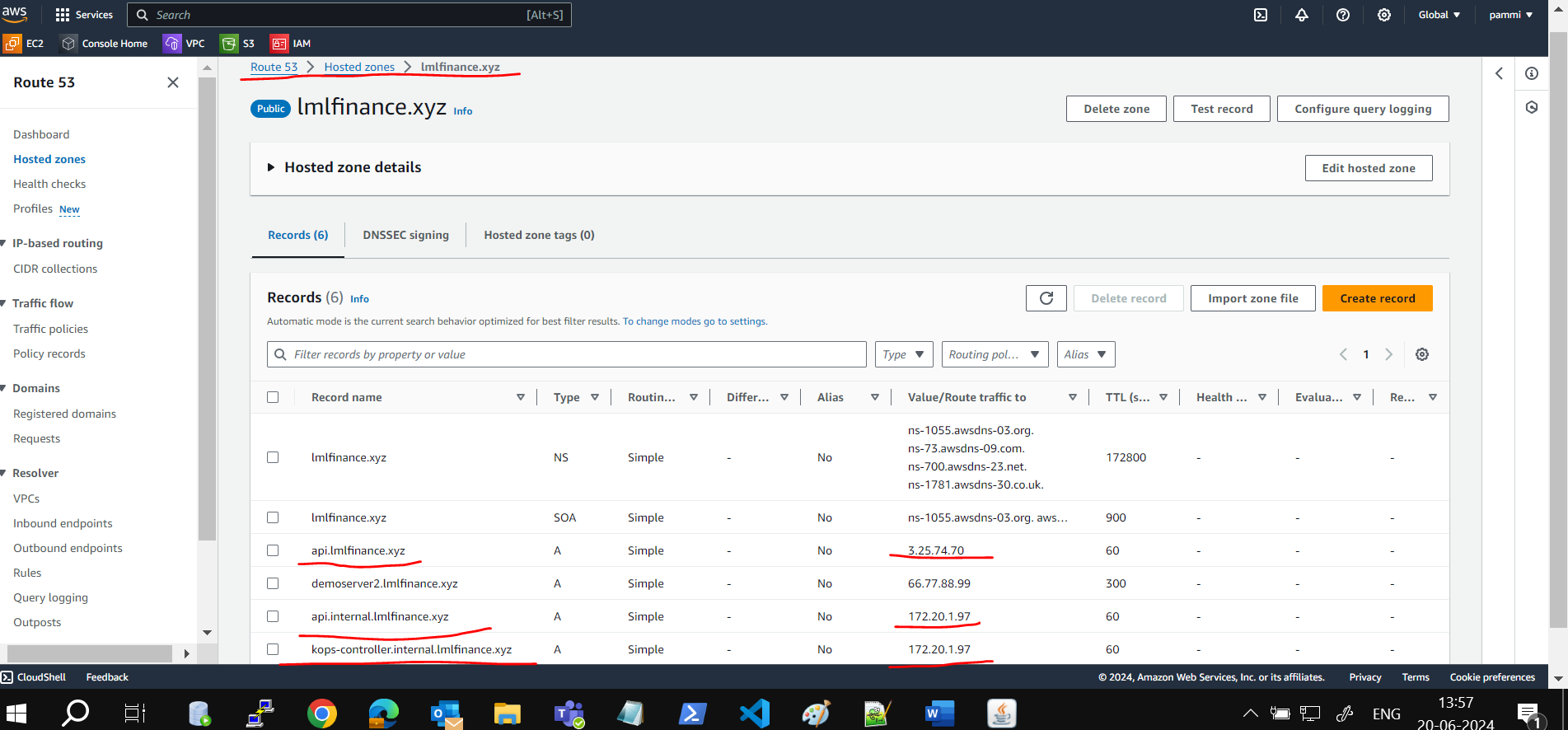


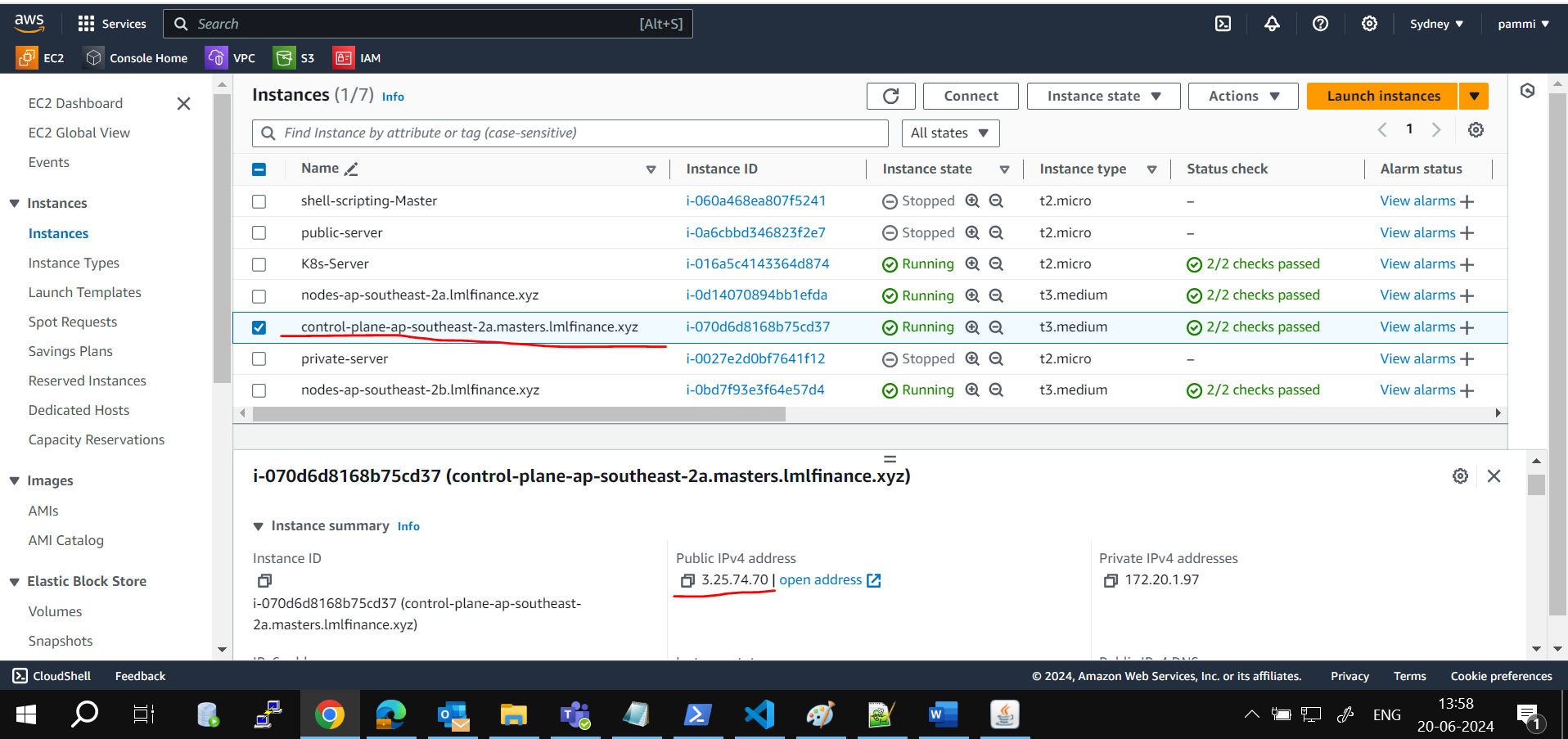




**Cluster creation Validation**





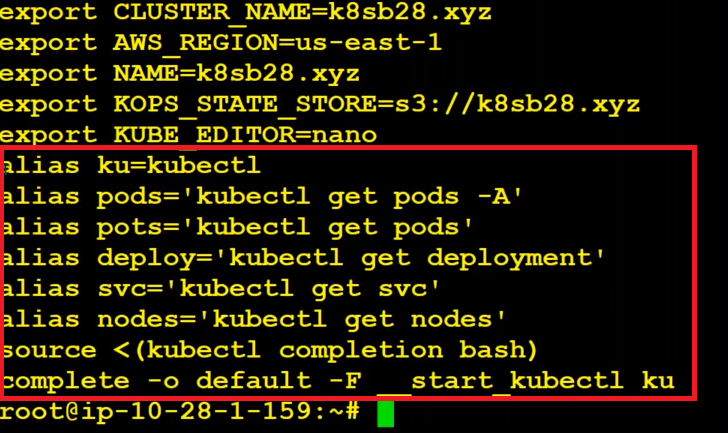


**Kubernetes shortcuts**

Add below shortcuts in **.bashrc** file. so going forward you don’t need to type full commands

# vim .bashrc

add below commands



Once above shortcuts are add refresh the env by using below command

# source .bashrc

Static pod 🡪 pods are not maintained by API server, pods which are available in manifest

When we can use static pods – High availability eventhogh api server down we needs to run our pods then we will create the pods in manifest

**Class 4**

1. Deploying K8s cluster on on-prem environment using **Rancher Kubernetes Engine** (RKE)
2. Run Rancher UI Application using Docker host
3. Run some sample applications on the cluster

RKE user created from IAM

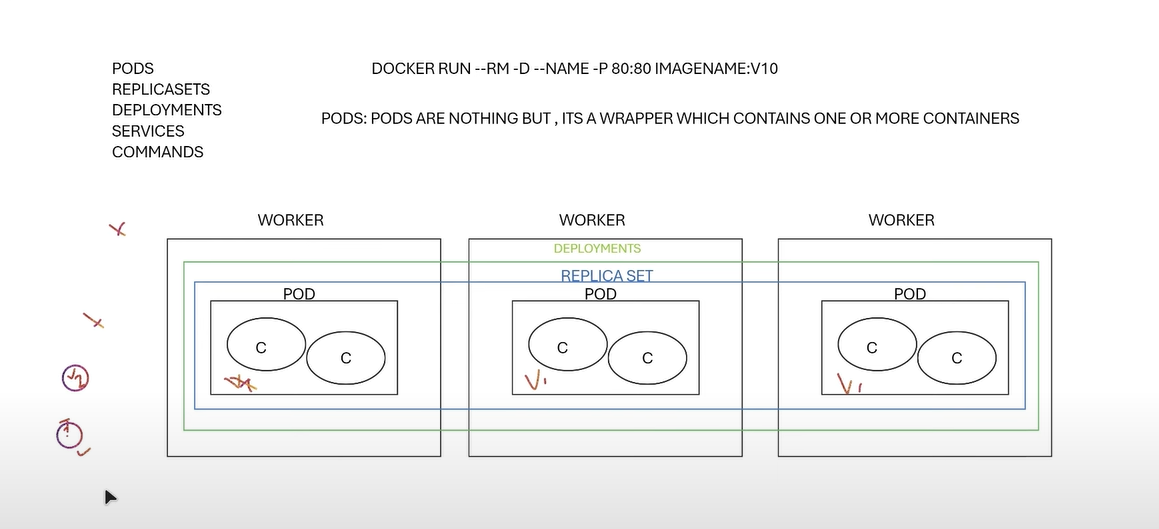
$ **export AWS\_ACCESS\_KEY\_ID=** ***AKIA6GBMGTSP3RHI6P5F***

$ **export AWS\_SECRET\_ACCESS\_KEY=** ***UBXboxRL9gdk1x7lDkgWV+6voqzqC8b7uBPOMAW6***

$ **export AWS\_DEFAULT\_REGION=*ap-southeast-2***

**Class 5**

**PODs-ReplciaSet-DeploymentOverview**

****

Pods are individual entities ( no one is there to take care of pods) if you delete the pod it will delete there is no backup planned

To protect pods 🡪 pods are deployed in multiple ways

* 1 pod can contain 1/more containers.

In case if the worker node 1 is down – pods in that worker node also down

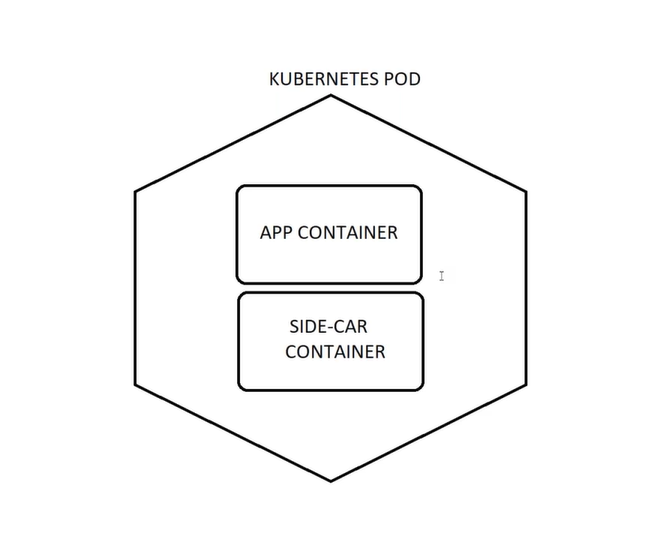
How can we protect the Pods by using Replicaset

**Replicaset**

Replicaset will take the pods high availability. So same pods will be replicated in multiple worker nodes (worker node 2 & 3) but dis-advantage is it can’t help in rolling updates to overcome this issue ( deployment came into picture)

**Deployment**

Deployment will help us for rolling updates



**Ways of resource deployment**

1. **Imperative format** (command execution)

* Kubectl run testpod1 –image nginx:latest

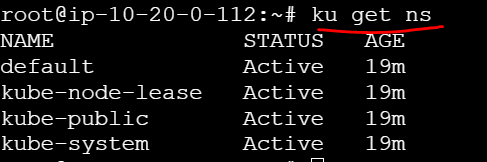
1. Declarative format (**using yaml/json**)

**Namespaces**

Namespaces are like folder.

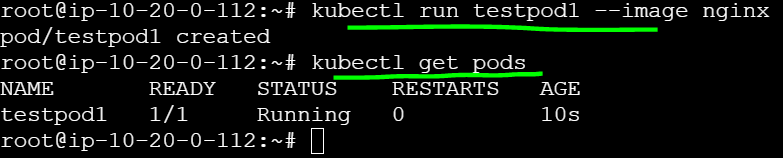
If we have multiple applications we can keep them in multiple namespaces

so we can apply the permissions on namespaces (who can have access/ how much cpu / memory/ network policies also can be applied)

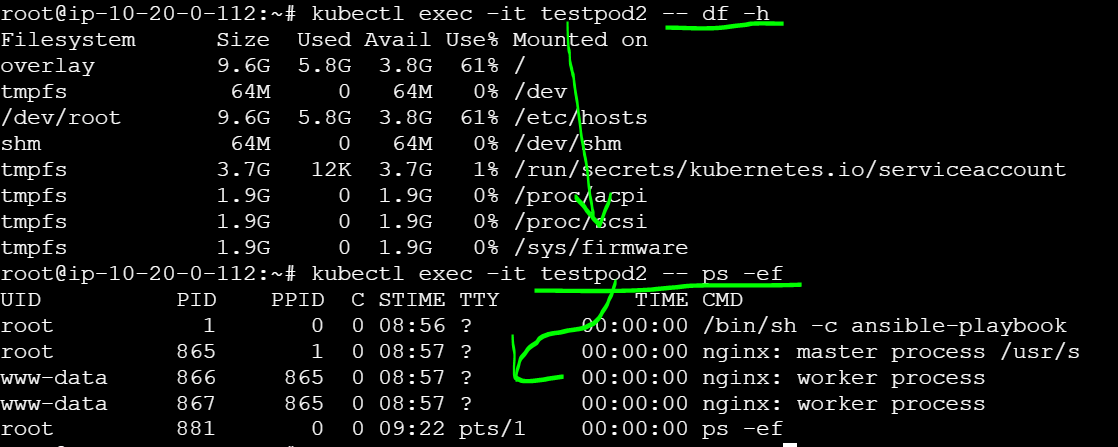


**Kube-system**  -- contain all management related information

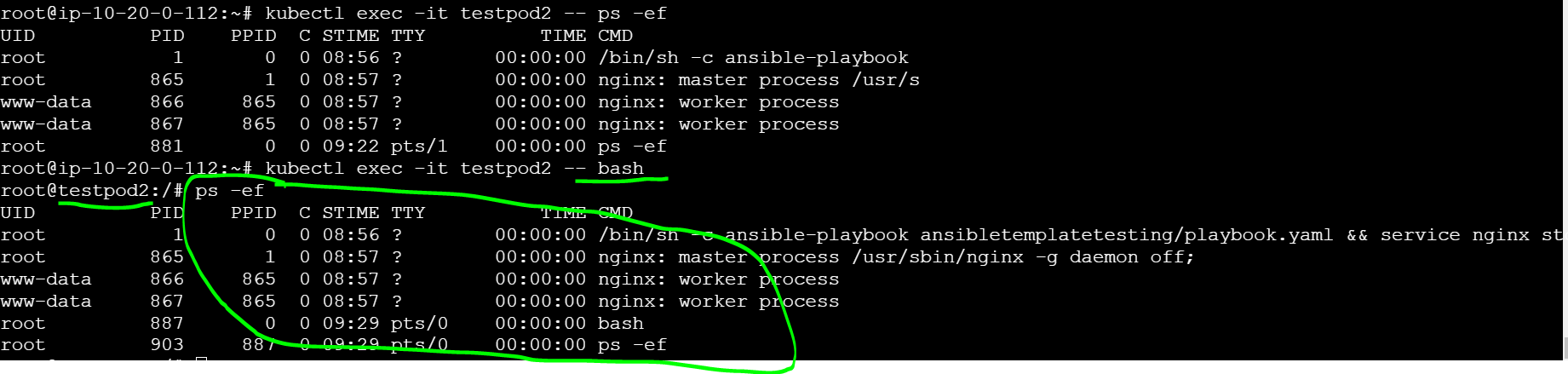
**Creation of pod**



**To get pod information from Main server**



**If you want to jump into pod**



**ReplicaSet**

#to secure the pod from failure/termination/deletion

apiVersion: apps/v1

kind: ReplicaSet

metadata:

  name: frontend

  labels:

    app: guestbook

    tier: frontend

spec:

  # modify replicas according to your case

  replicas: 3

  selector:

    matchLabels:

      tier: frontend

  template:

    metadata:

      labels:

        tier: frontend

    spec:

      containers:

      - name: app

        image: sreeharshav/rollingupdate:v1

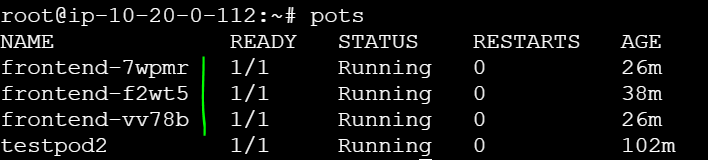
#linux command to split screen

tmux --> ctrl b + shift "  (you will see 2 windows in 1 screen)

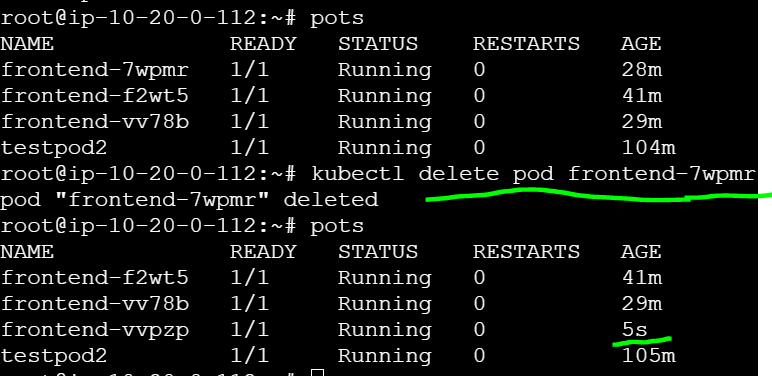
#how to move the cursor in this case

ctrl v + up arrow / ctrl v + down arrow

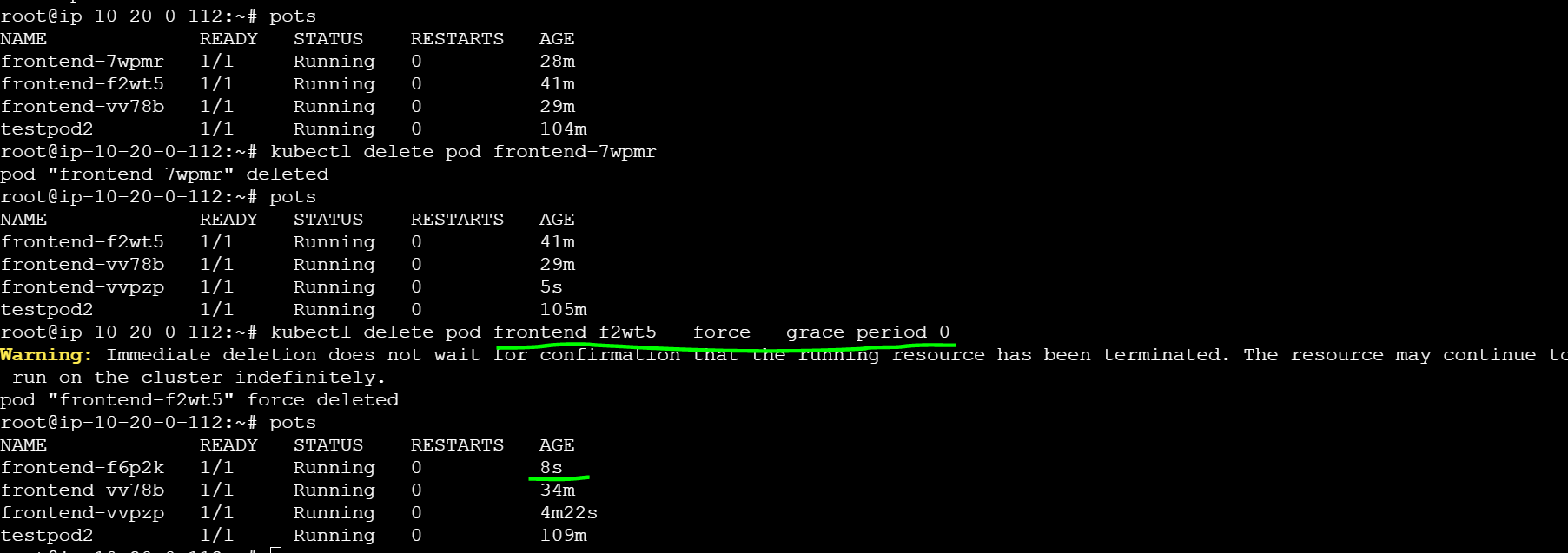
watch -n 1 kubectl get pods



If you delete the pod, it will automatically create new pod

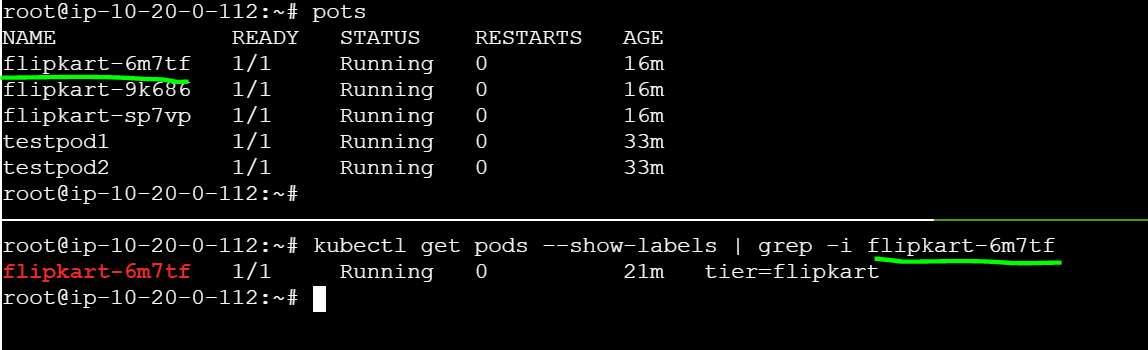


if you are facing any issue while deleting the pod use below command



**Label**

kubectl get pods --show-labels | grep -i flipkart-6m7tf

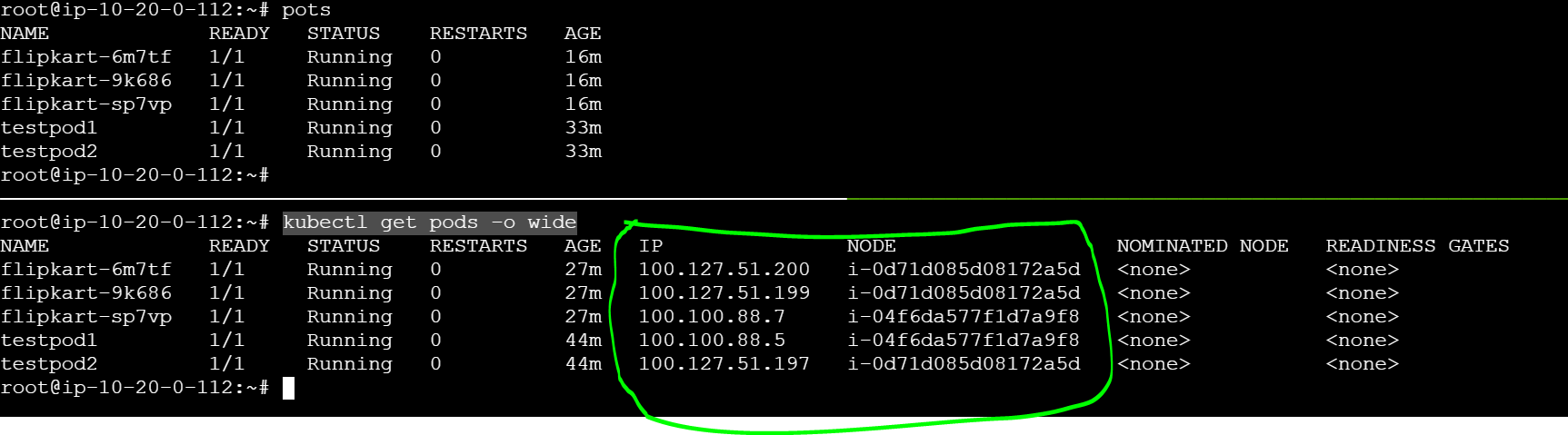


**Change pod label**

kubectl label pod testpod1 tire=flipkart

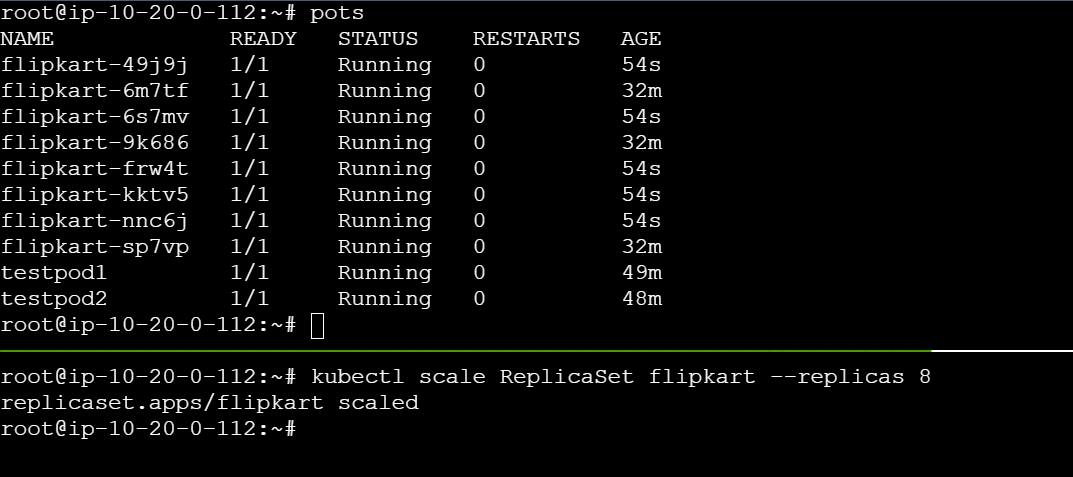


kubectl get pods -o wide

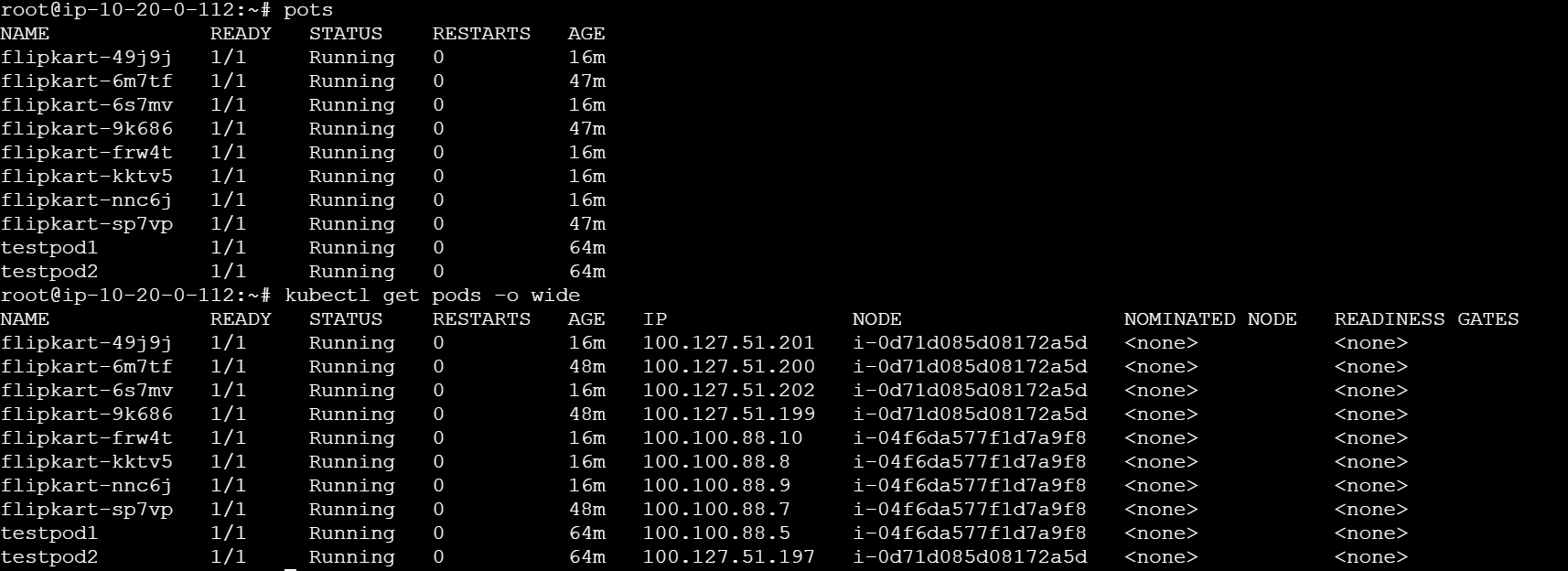


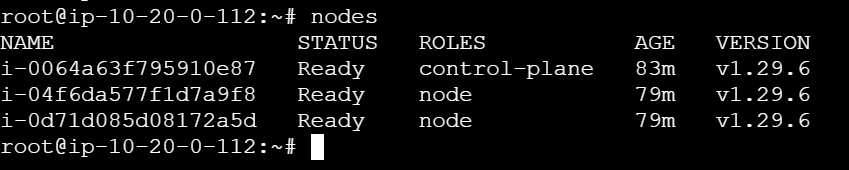
**Scale pods**





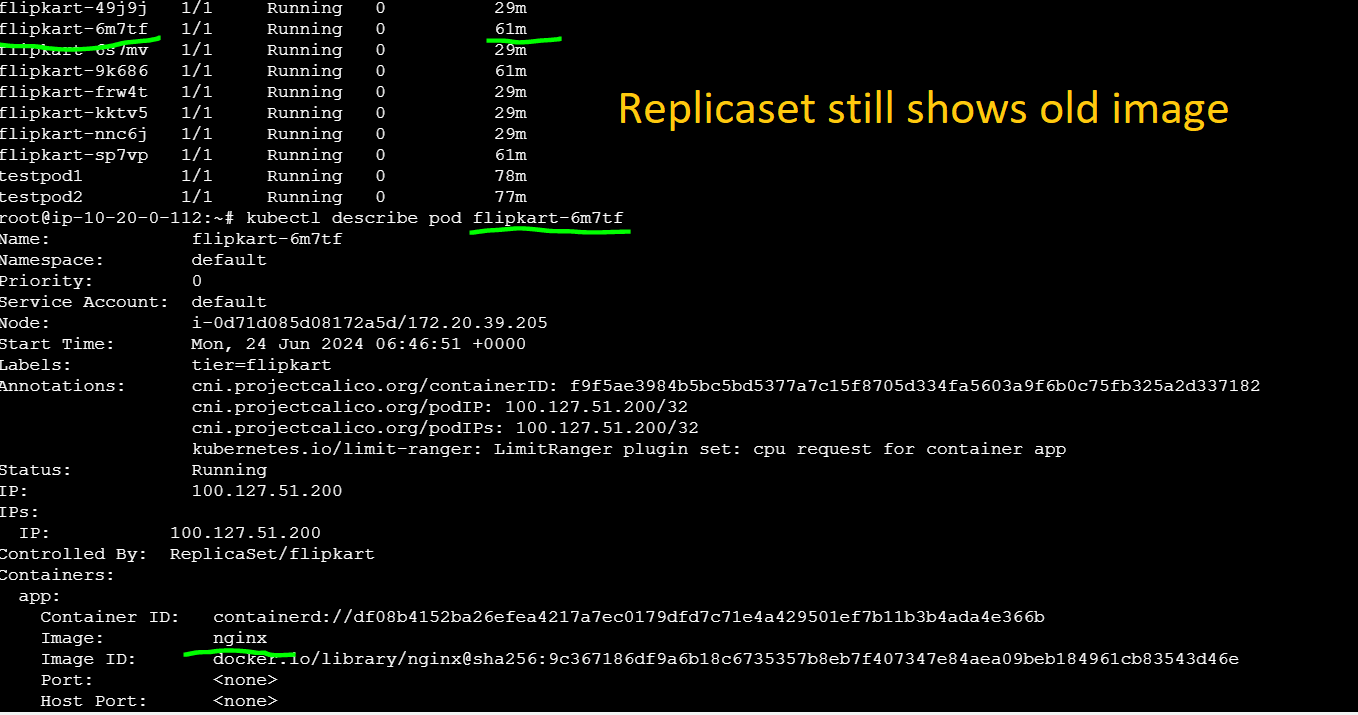
**kubectl get pods -o wide**



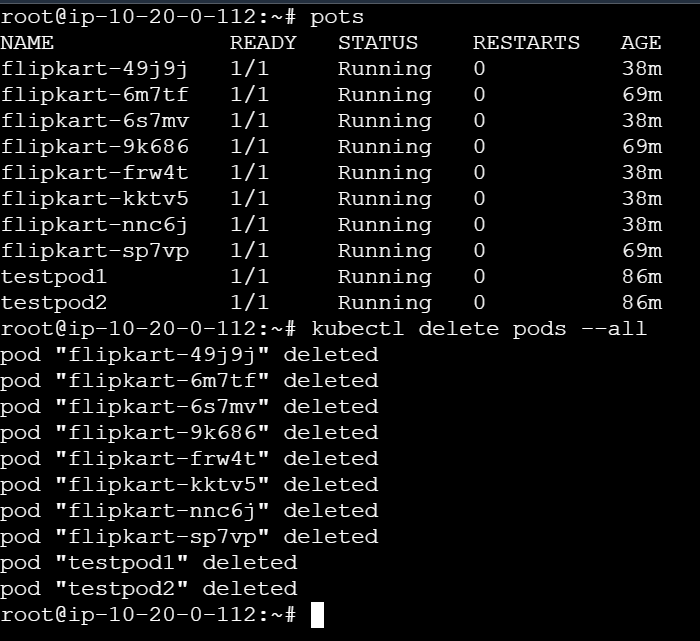


**Dis-advantage of Replicaset (it will not perform automatic updation)**

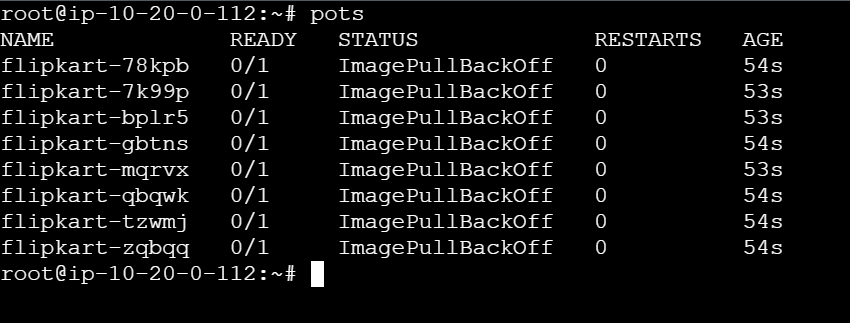


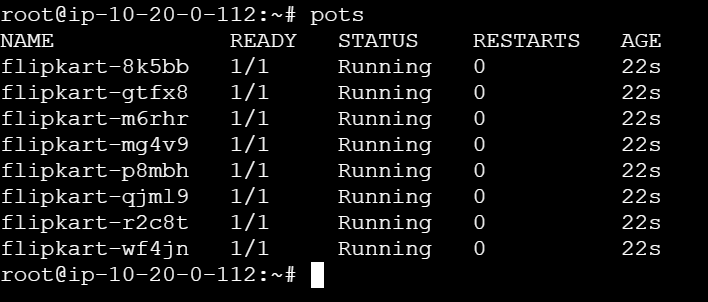
****

**Delete pods**

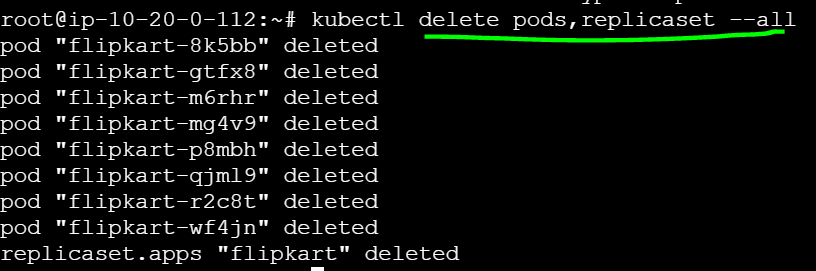


**Error: Incorrect image name provided**



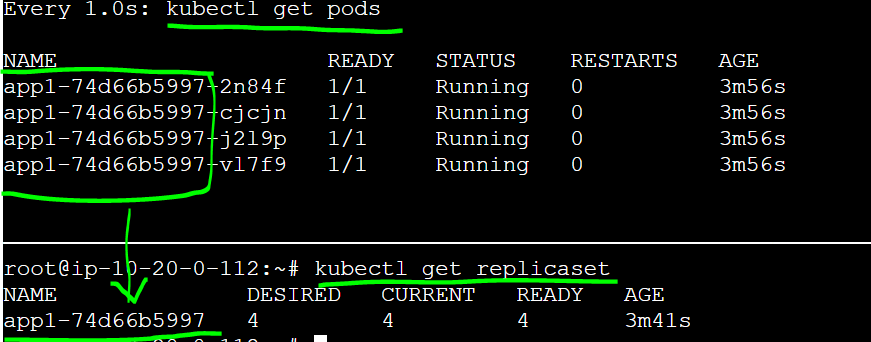


**kubectl delete pods,replicaset --all**

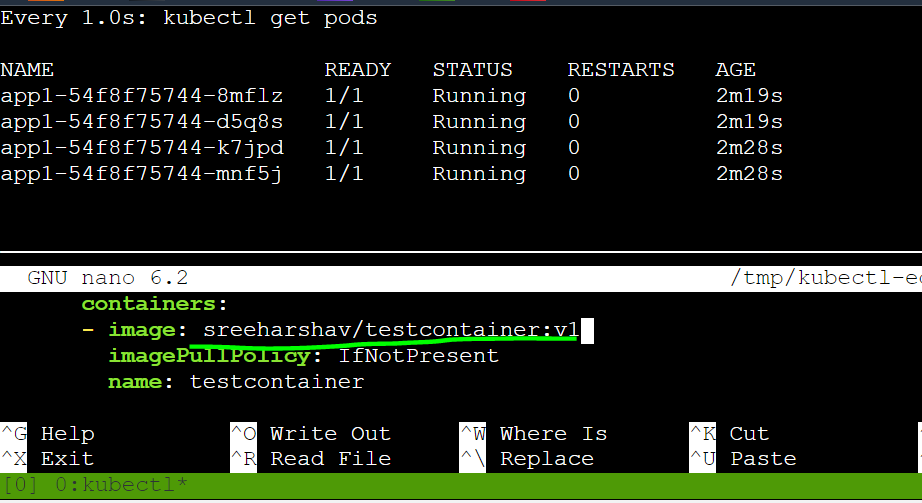


**To overcome Replicaset dis-advantage 🡪 deployment came into the picture**

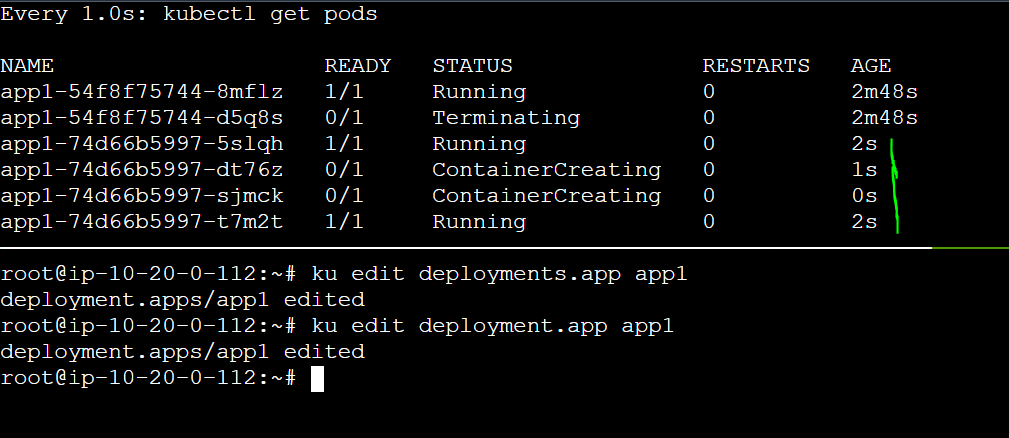
**kubectl get replicaset**



**kubectl edit deployments.app app1**

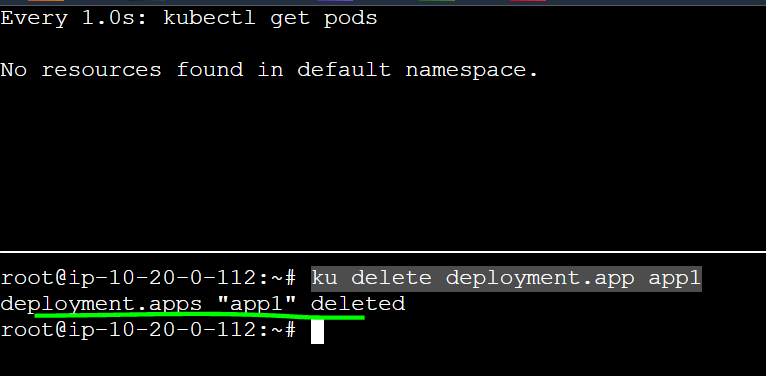


**Note: once image name update, some pods are automatically tries to use new image if it is success then only remaining pods will update there image.**

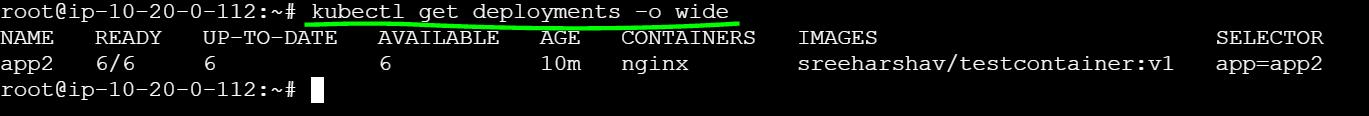


**Delete deployment**

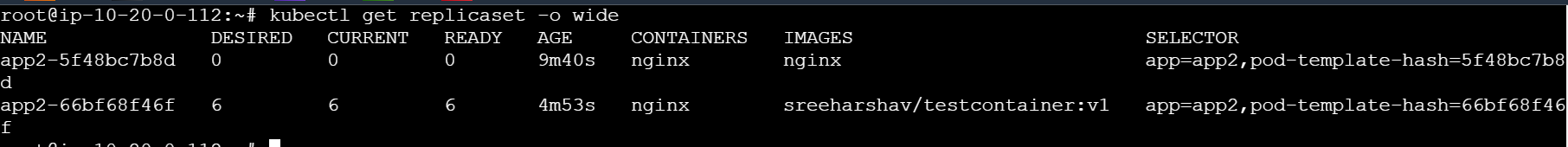
**kubectl delete deployment.app app1**



**kubectl get deployments -o wide**



**kubectl get replicaset -o wide**



**kubectl get pods --show-labels**

