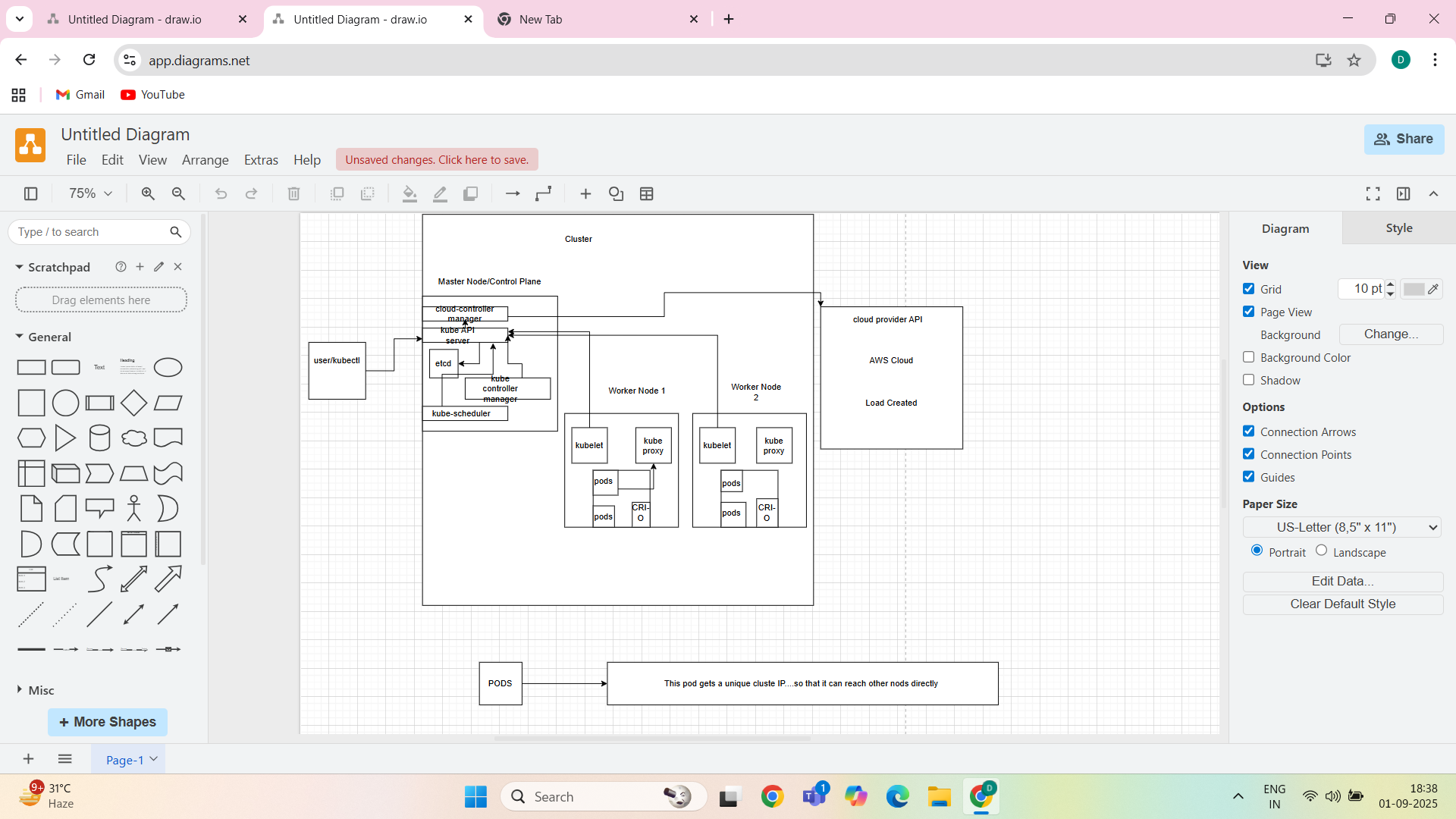
Kubernetes Architecture



Master Node (Control Plane)

kube-apiserver – API entry point

etcd – stores cluster data

kube-scheduler – schedules pods

kube-controller-manager – controllers (node, job, etc.)

cloud-controller-manager

Worker Nodes

kubelet – manages pods on the node

kube-proxy – networking (Service communication)

Container Runtime (Docker, containerd)

Pod-In **Kubernetes**, a **Pod** is the **smallest and simplest deployable unit**.

Pod yaml File-

apiVersion: v1 # API version for Pod resource

kind: Pod # Type of Kubernetes object

metadata:

name: my-pod # Name of the pod

labels:

app: myapp # Label assigned to pod

spec:

containers: # List of containers in the pod

- name: my-container # Name of the container

image: nginx # Docker image to use

ports:

- containerPort: 80 # Port exposed by container

Commands to run on Killercoda:

# create pod

kubectl apply -f pod.yaml

# check pods

kubectl get pods

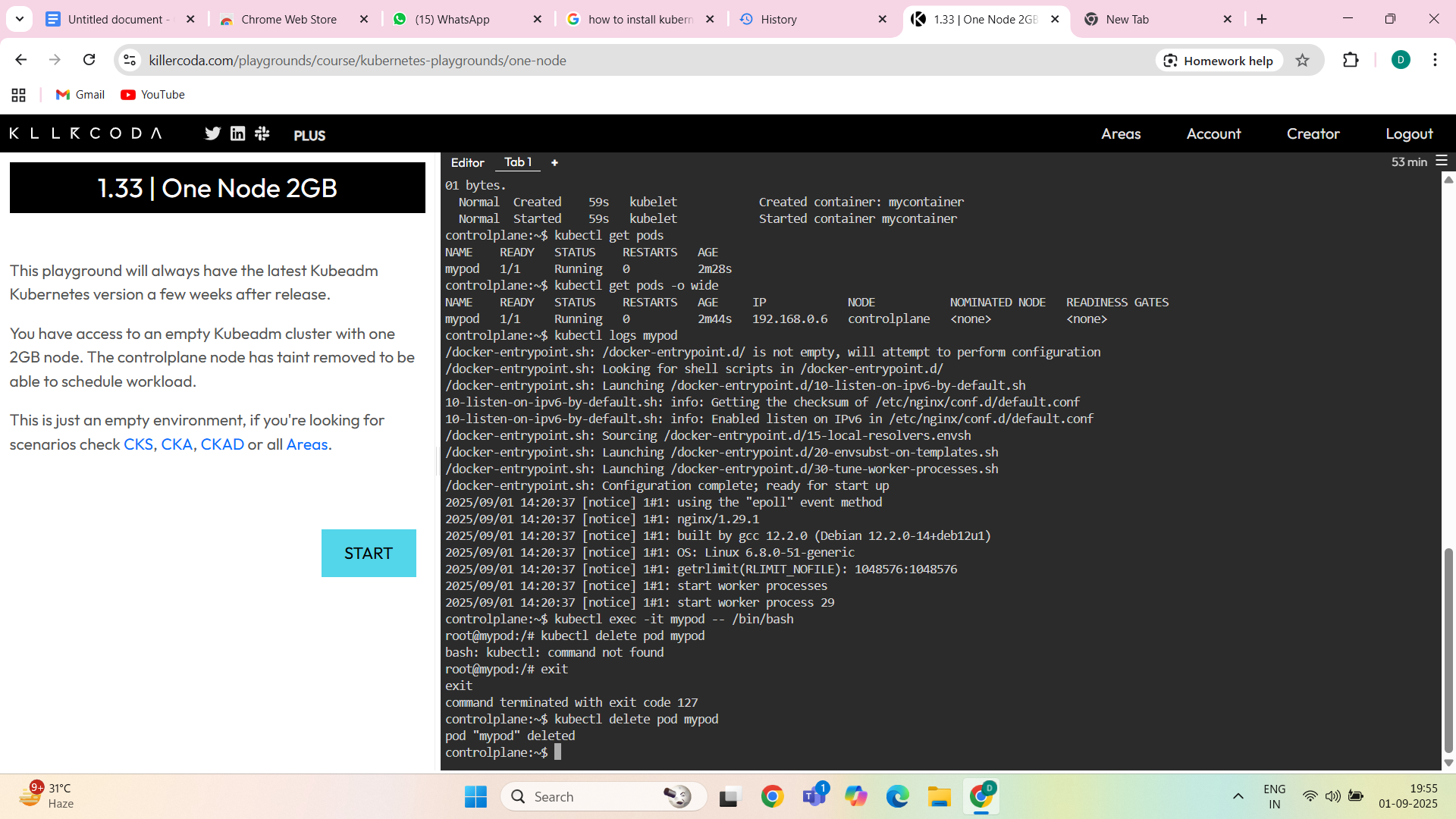
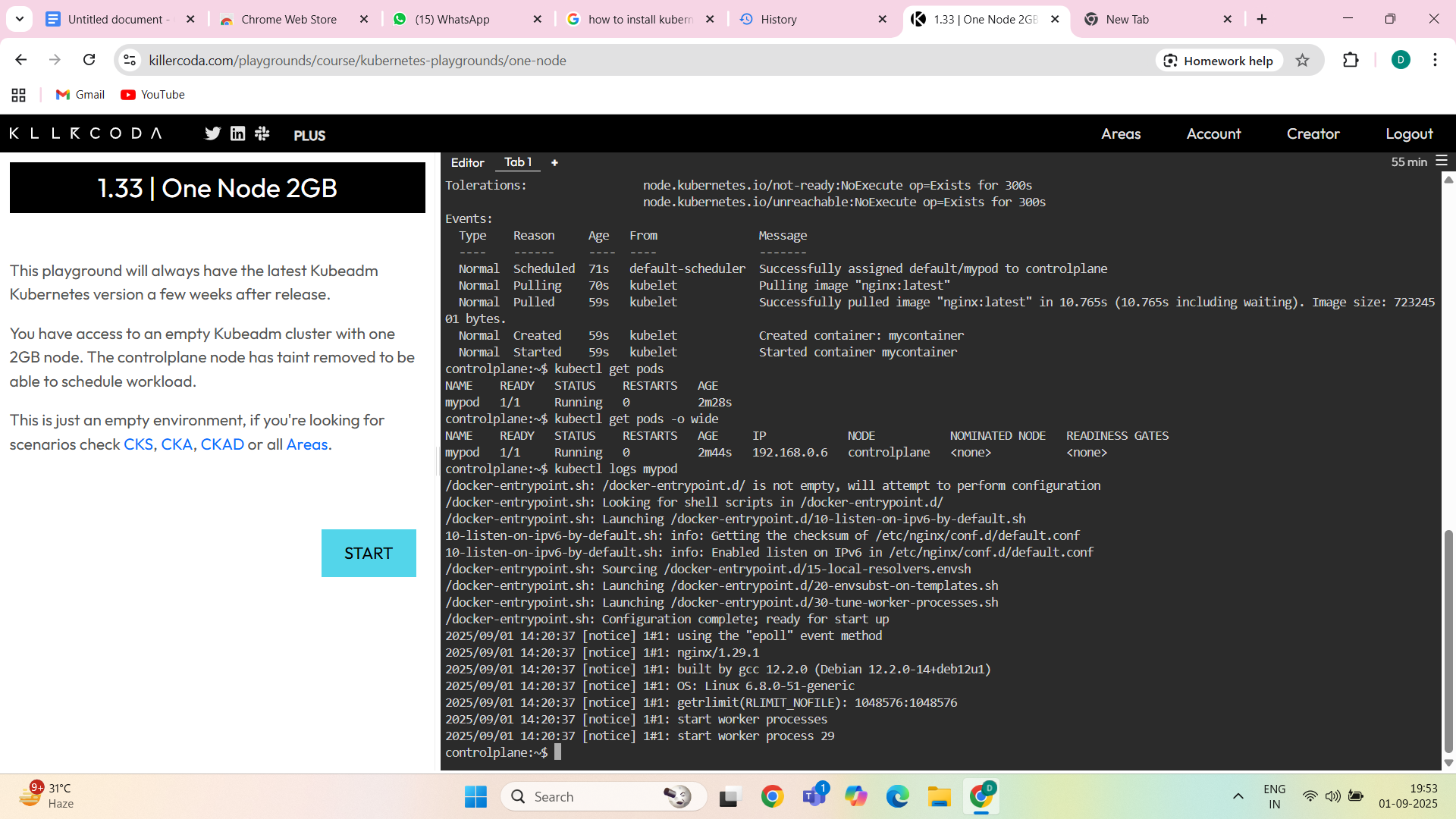
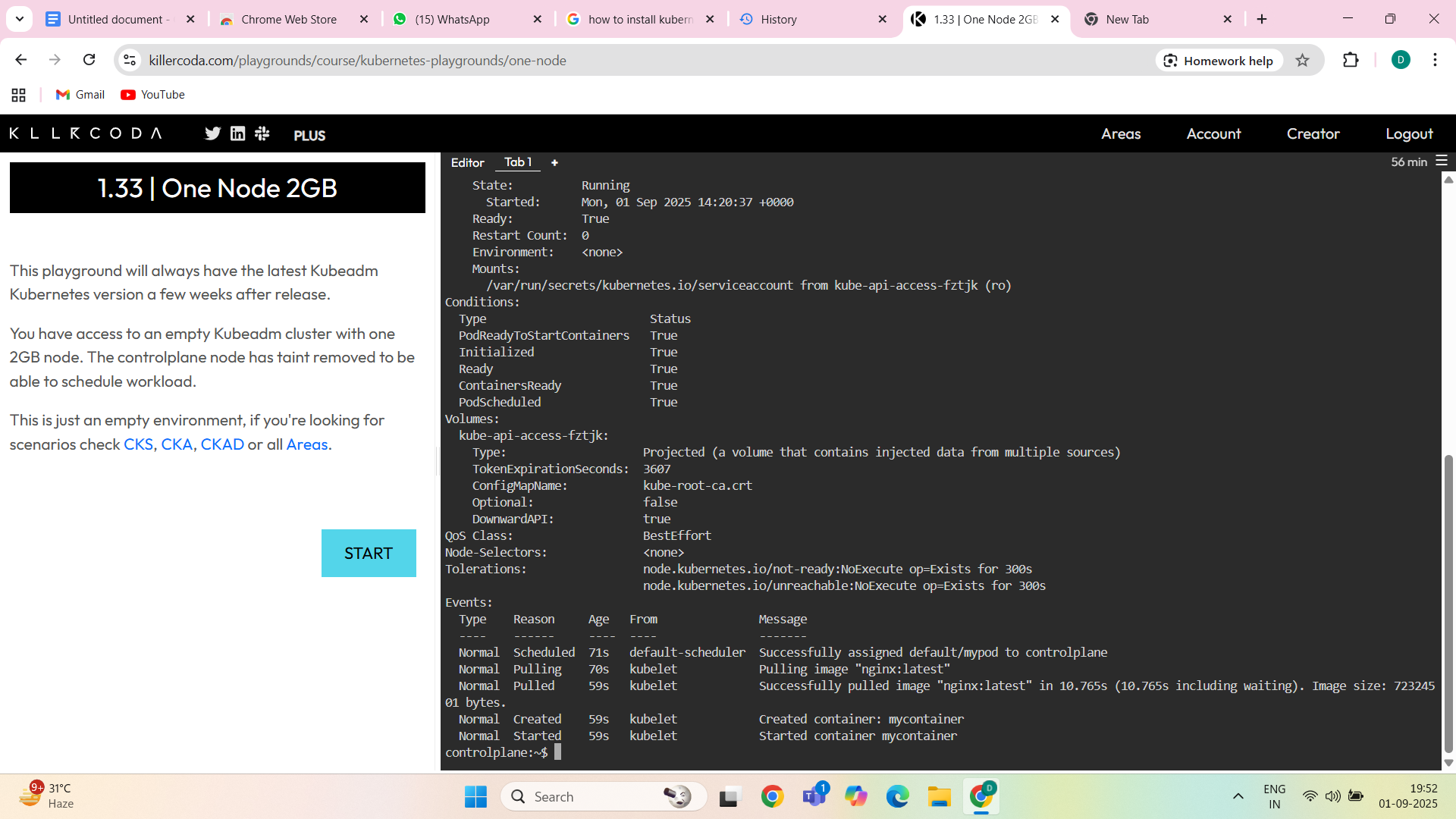
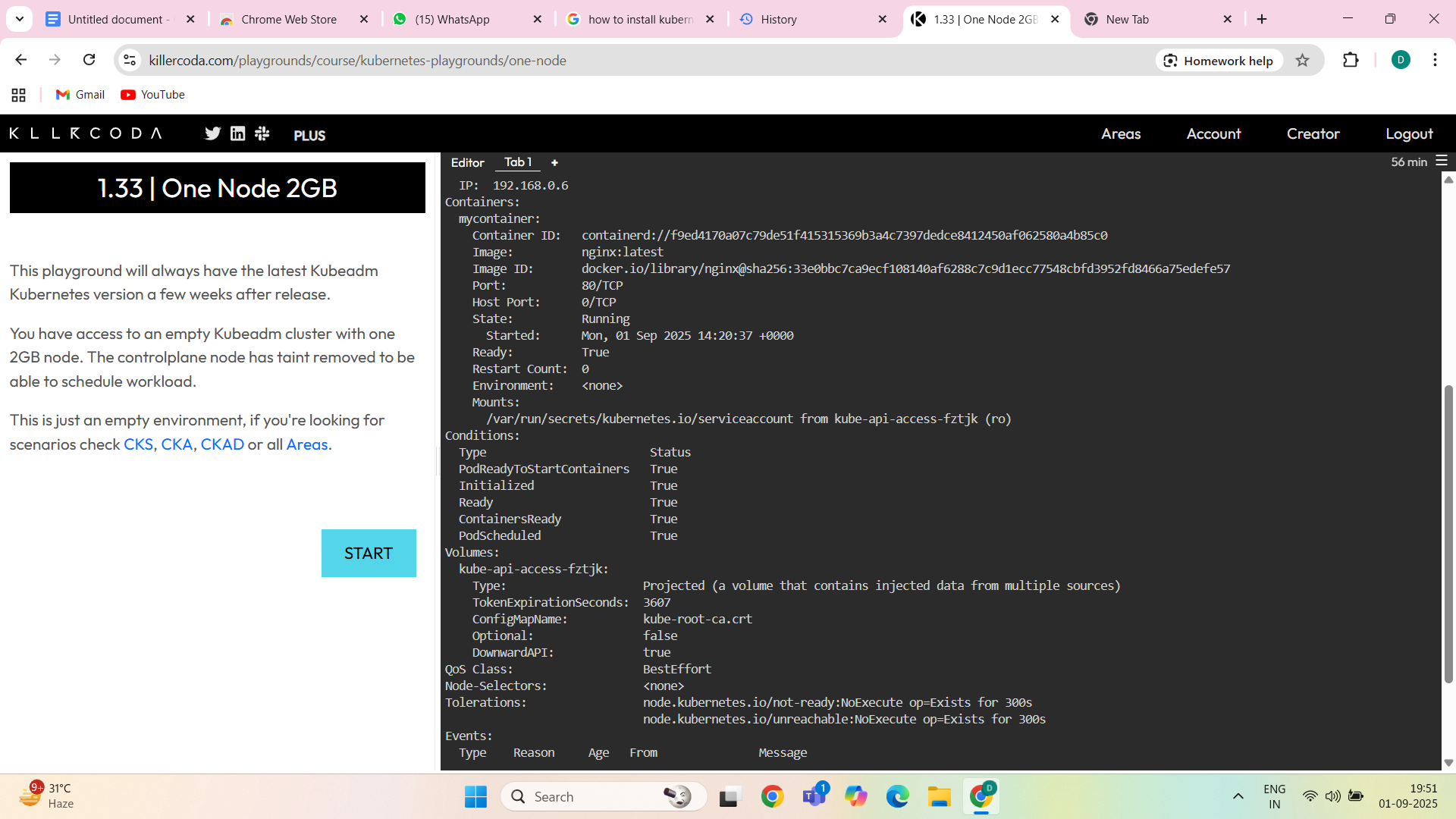
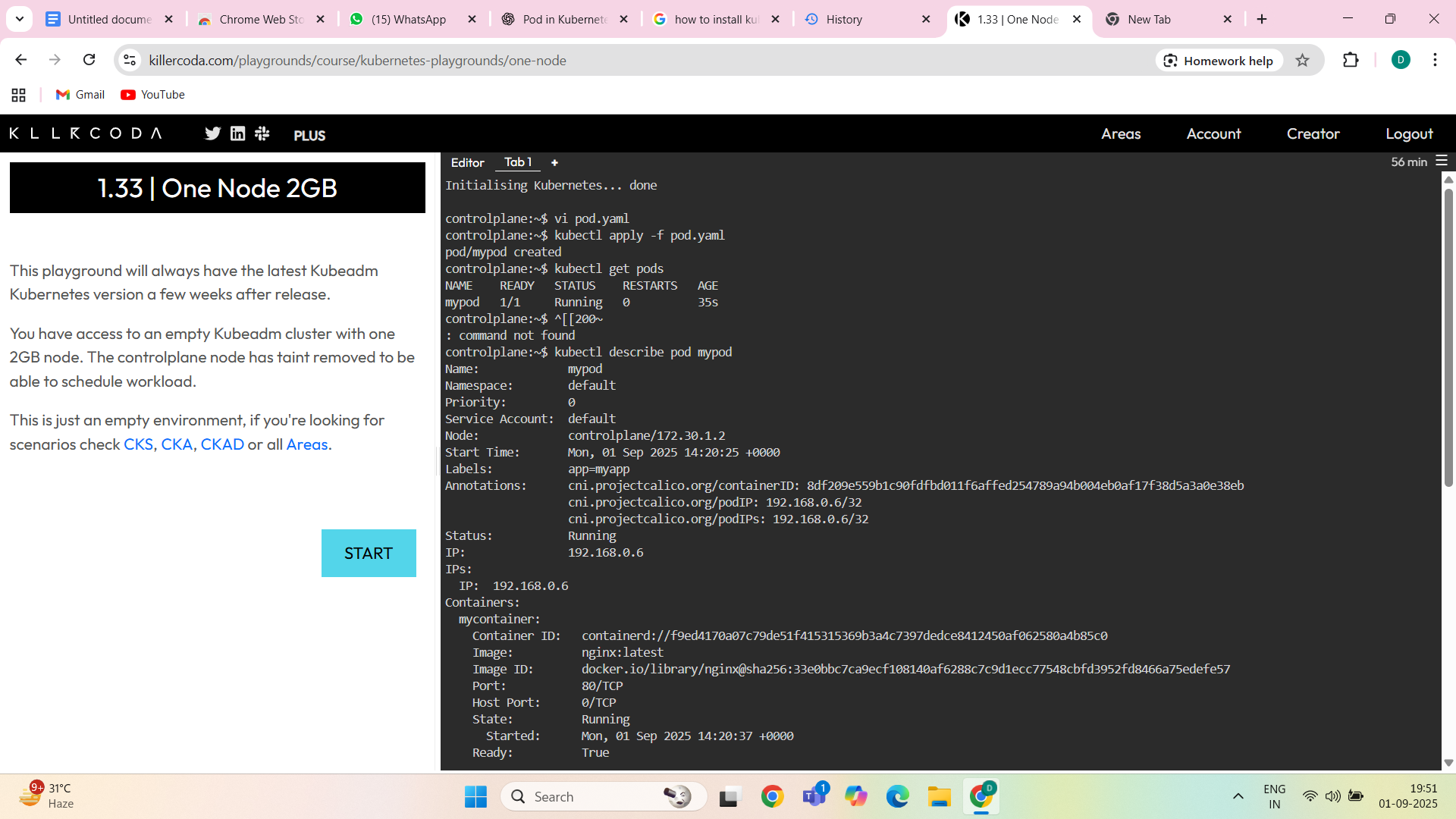
# describe pod

kubectl describe pod mypod

# delete pod

kubectl delete pod mypod

kubectl exec -it mypod -- /bin/bash



REPLICASET-A **ReplicaSet** maintains a stable set of replica Pods running at any given time.

If a Pod fails, crashes, or is deleted, the ReplicaSet automatically **creates a new Pod** to replace it.

Replicaset yaml file-

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: my-rs

namespace: sinchan

spec:

replicas: 3

selector:

matchLabels:

app: myapp

template:

metadata:

labels:

app: myapp

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

commands-

vi replicaset.yaml

kubectl get rs

kubectl create ns sinchan

kubectl get rs

kubectl apply -f replicaset.yaml

kubectl get rs

kubectl describe rs my-rs

kubectl describe rs sinchan

mkdir sinchan

cd sinchan

vi replicaset.yaml

ls

kubectl apply -f replicaset.yaml

kubectl describe rs sinchan

kubectl get rs

kubectl describe rs my-rs

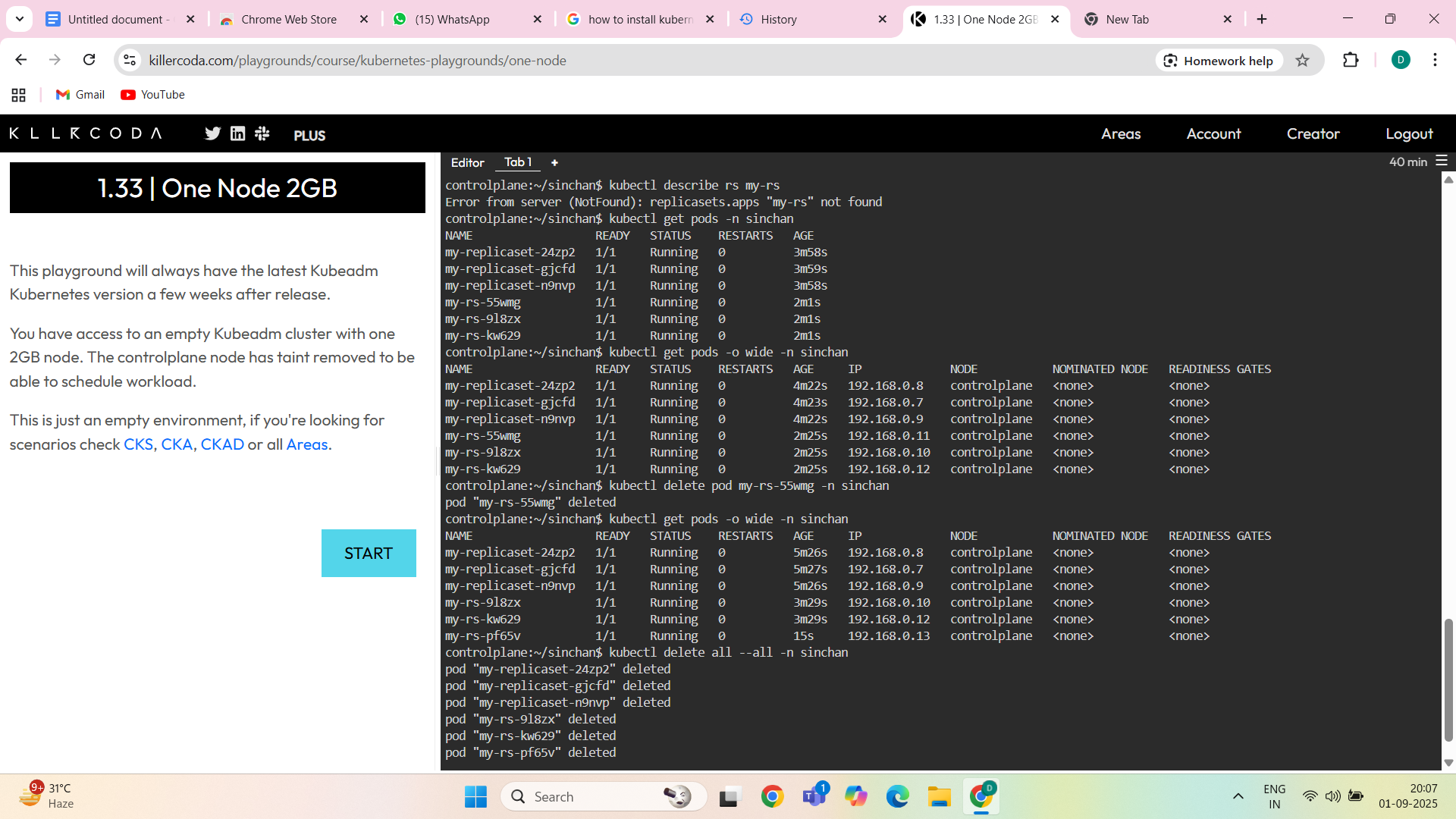
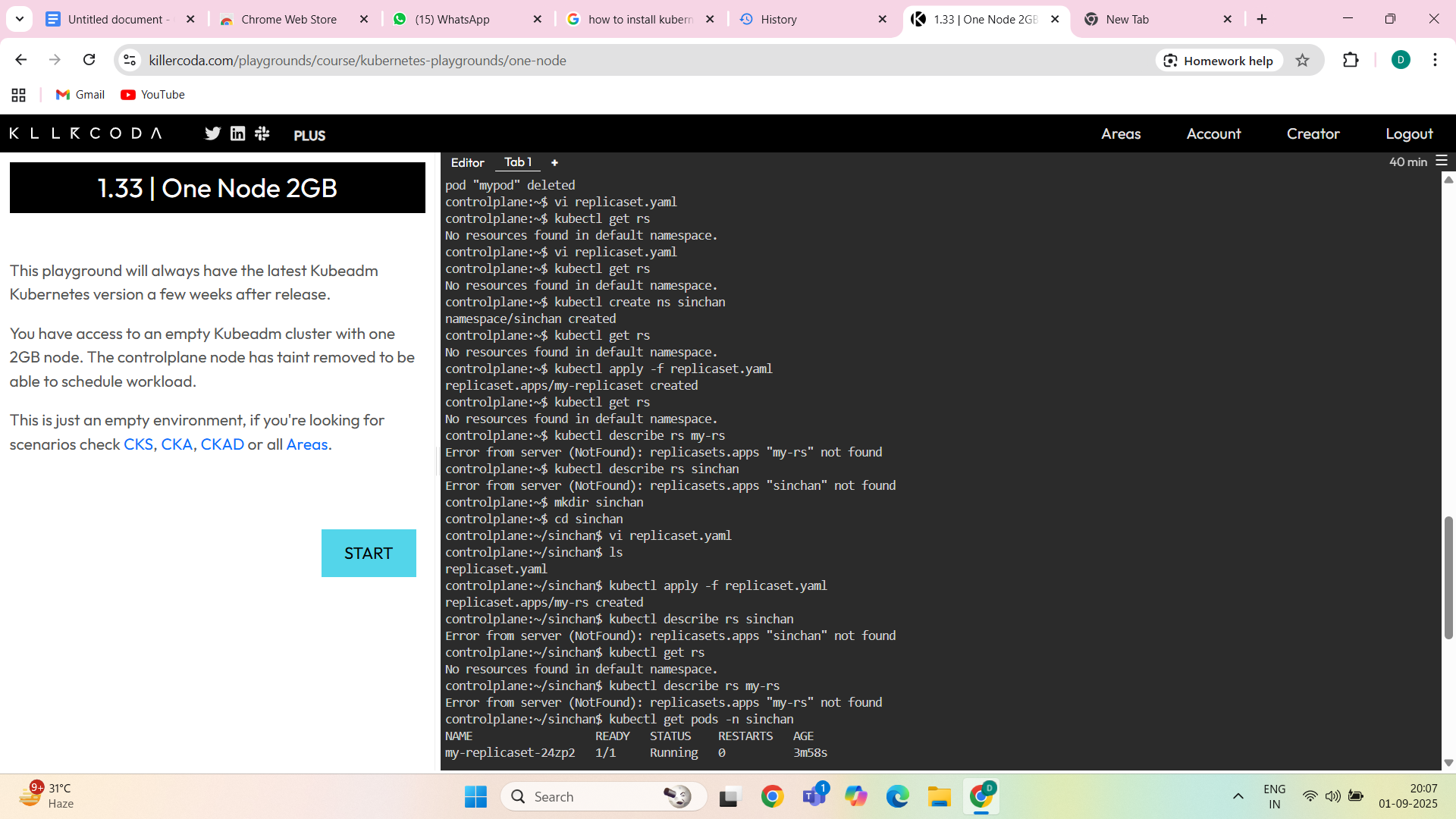
kubectl get pods -n sinchan

kubectl get pods -o wide -n sinchan

kubectl delete pod my-rs-55wmg -n sinchan

kubectl get pods -o wide -n sinchan

kubectl delete all --all -n sinchan



DEPLOYMENT-A **Deployment** manages **ReplicaSets** and provides **declarative updates** for Pods and ReplicaSet.

Deployment yaml file-

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-deployment

namespace: sinchan

spec:

replicas: 3

selector:

matchLabels:

app: myapp

template:

metadata:

labels:

app: myapp

spec:

containers:

- name: nginx-container

image: nginx

ports:

- containerPort: 80

Deployment commands-

vi deployment.yaml

kubectl apply -f deployment.yaml

cubectl get deployment -n sinchan

kubectl get deployment -n sinchan

kubectl get all -n sinchan

kubectl describe deployment my-deployment

kubectl get all -n sinchan

kubectl describe deployment my-deployment

kubectl rollout status deployment/my-deployment

kubectl describe deployments my-deployment

kubectl describe deployments

kubectl describe deployments my-deployment -n sinchan

kubectl rollout status deployment/my-deployment

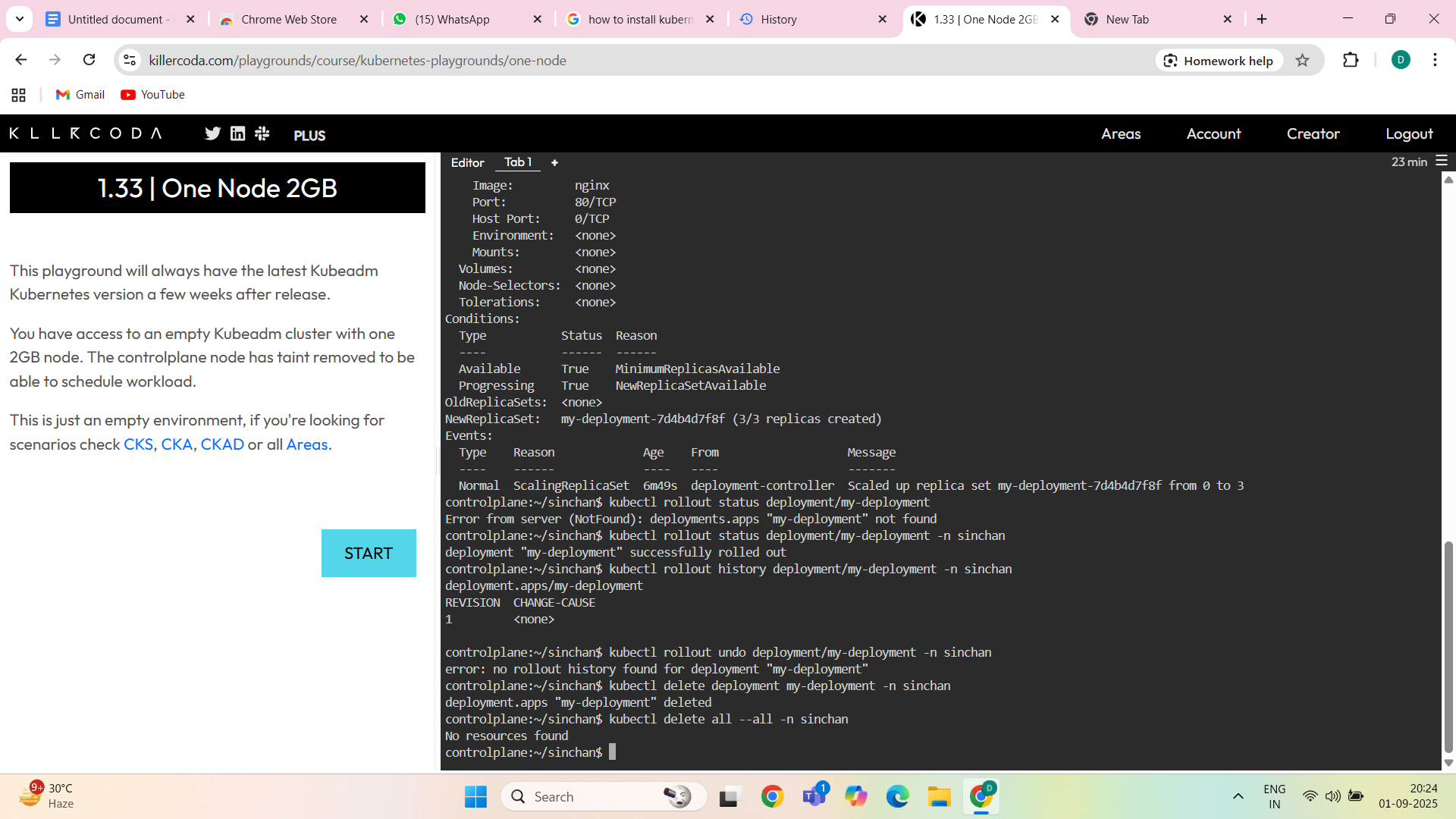
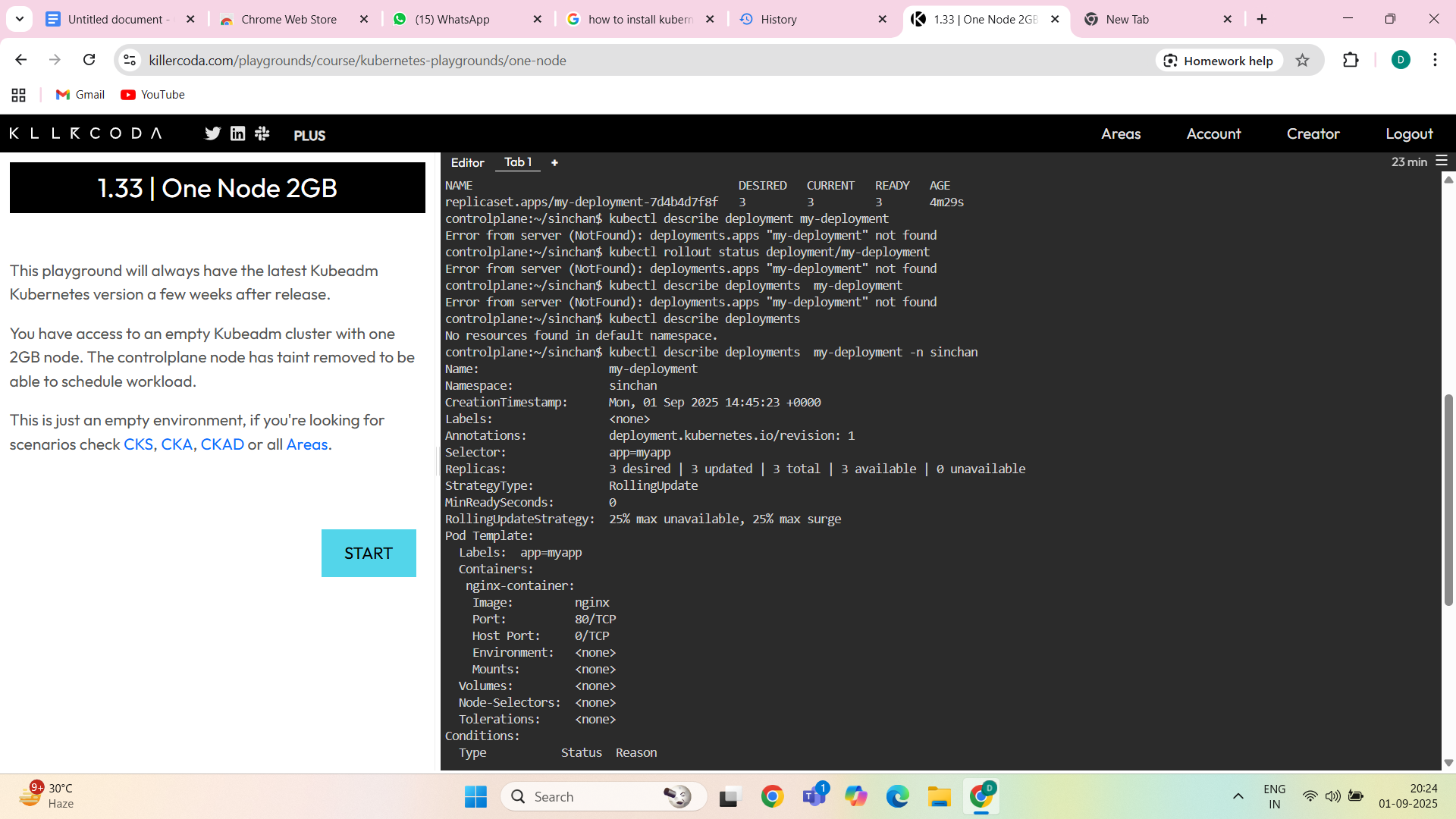
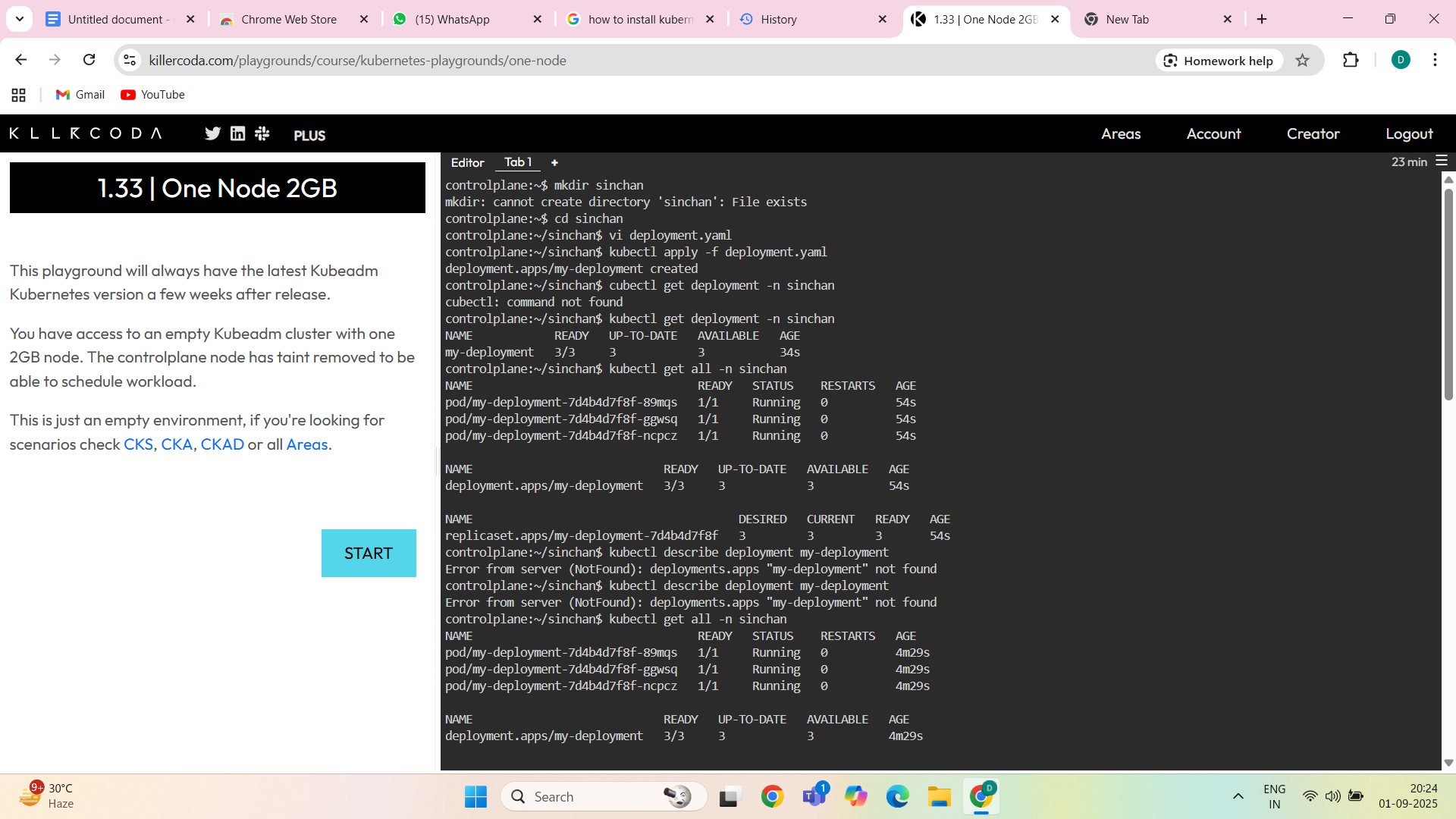
kubectl rollout status deployment/my-deployment -n sinchan

kubectl rollout history deployment/my-deployment -n sinchan

kubectl rollout undo deployment/my-deployment -n sinchan

kubectl delete deployment my-deployment -n sinchan

kubectl delete all --all -n sinchan



CLUSTERIP-ClusterIP is the **default Service type** in Kubernetes.

It exposes the Service **inside the cluster only** (not accessible from outside).

Kubernetes assigns a **virtual IP (ClusterIP)** that other Pods in the cluster can use to communicate with it.

Traffic to this IP is load balanced across the backend Pods selected by the Service.

ClusterIp yaml file-

apiVersion: v1

kind: Service

metadata:

name: ipl-service

namespace: sinchan

spec:

selector:

app: ipl

ports:

- protocol: TCP

port: 80 #svc port number and its our wish

targetPort: 3000 # container port number and we shouldn't change it

type: ClusterIP

commands-

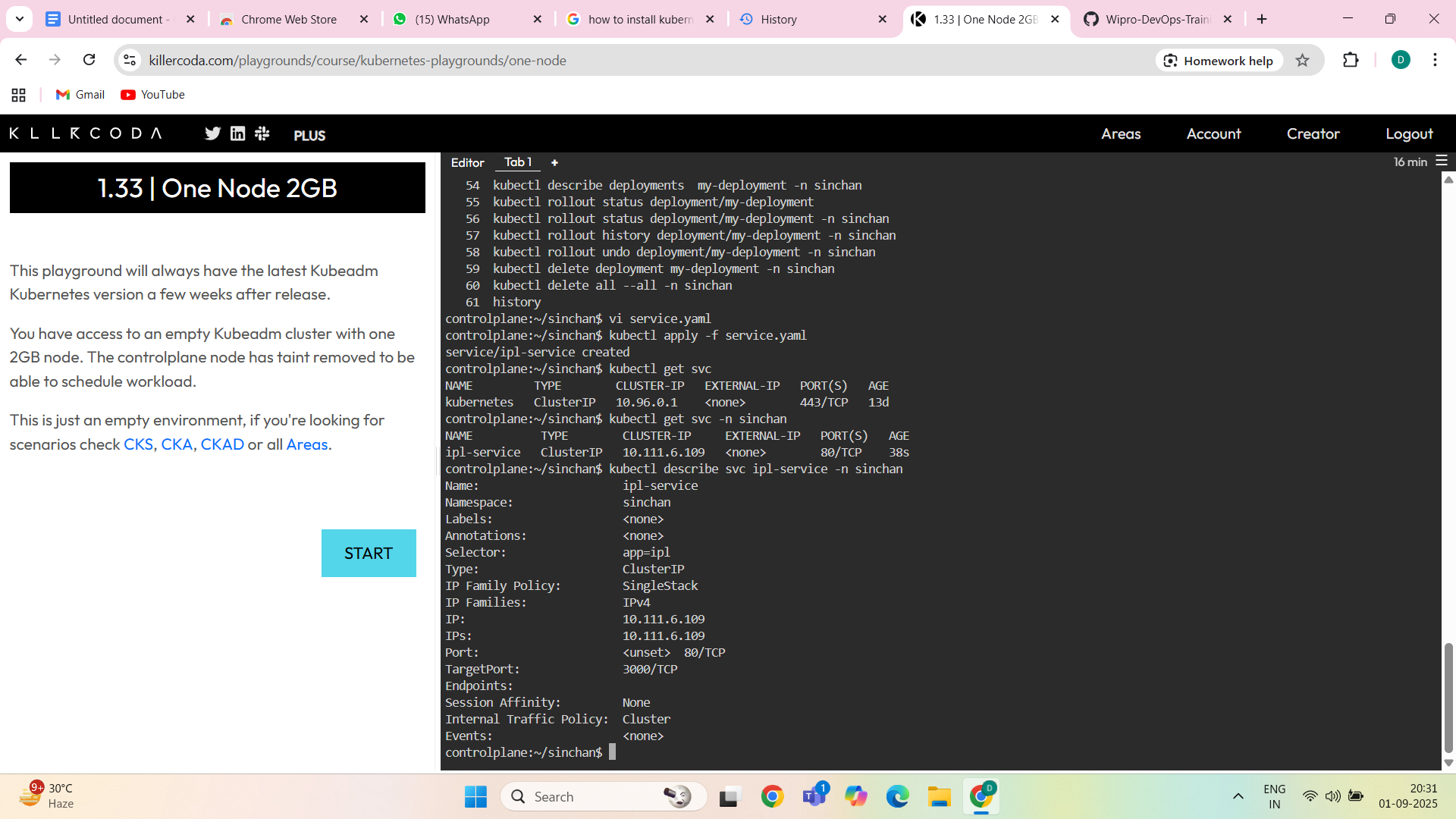
vi service.yaml

kubectl apply -f service.yaml

kubectl get svc

kubectl get svc -n sinchan

kubectl describe svc ipl-service -n sinchan



NODEPORT-NodePort is a **Service type** in Kubernetes that exposes your application **outside the cluster** using a port on each worker node.

Kubernetes allocates a **port between 30000–32767** on every Node in the cluster.

Any request coming to <NodeIP>:<NodePort> will be forwarded to the Service, and then to the backend Pods.

Nodeport yaml file-

apiVersion: v1

kind: Service

metadata:

name: ipl-service

namespace: sinchan

spec:

selector:

app: ipl

ports:

- protocol: TCP

port: 80 #svc port number and its our wish

targetPort: 3000 # container port number and we shouldn't change it

nodePort: 30080

type: NodePort

commands-

vi nodeport.yaml

ls

kubectl get all

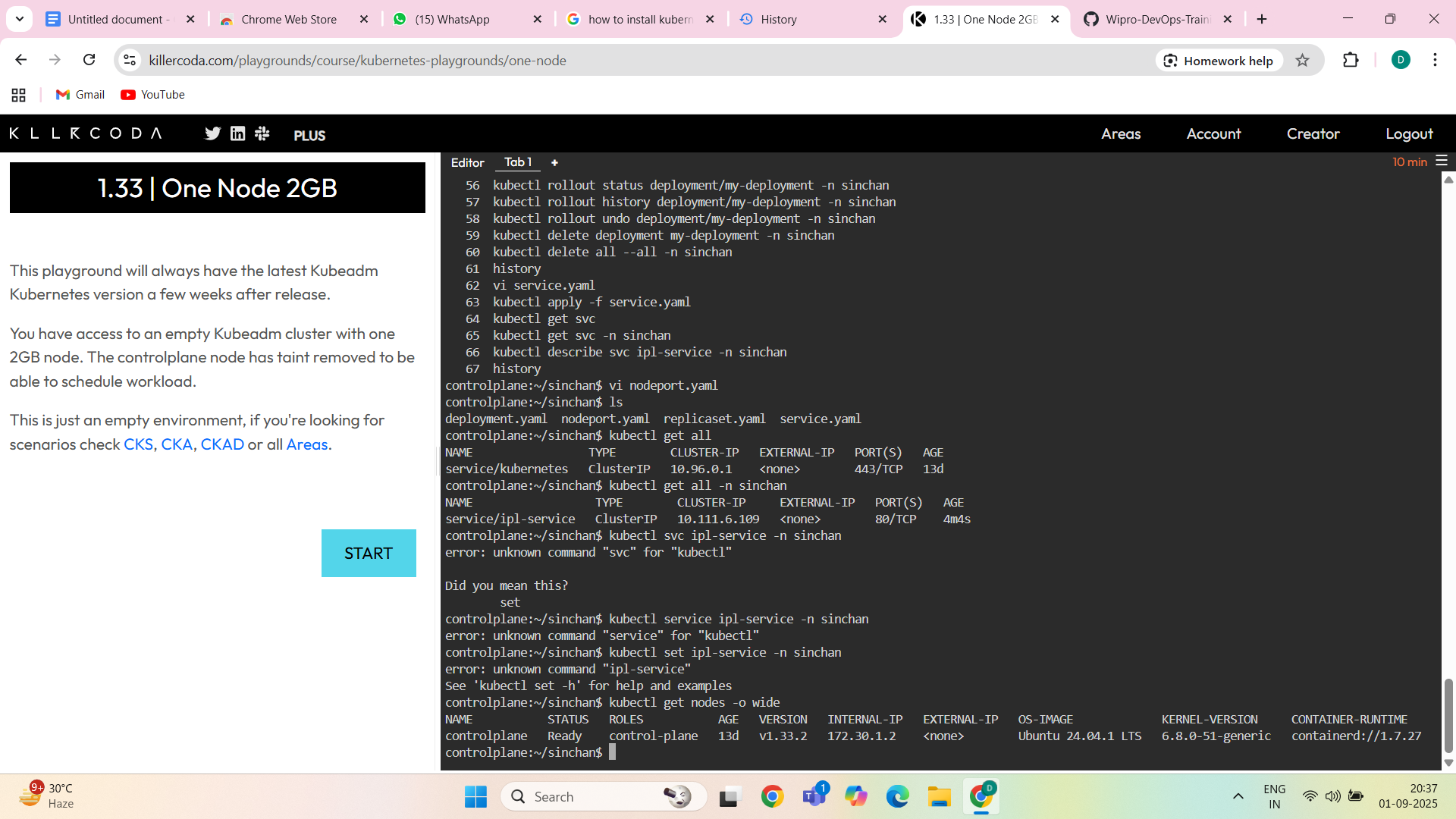
kubectl get all -n sinchan

kubectl svc ipl-service -n sinchan

kubectl service ipl-service -n sinchan

kubectl set ipl-service -n sinchan

kubectl get nodes -o wide



LOADBALANCER-In **Kubernetes**, a **LoadBalancer** in service.yaml is a **Service type** that exposes your application **to the outside world** using the cloud provider’s load balancer (like AWS ELB, Azure Load Balancer, GCP LB, etc.).

loadbalancer-apiVersion: v1

kind: Service

metadata:

name: lb-svc

namespace: sinchan

spec:

selector:

app: myapp

ports:

- port: 80

targetPort: 80

type: LoadBalancer

Commands-

kubectl apply -f service.yaml

kubectl get svc

kubectl describe svc my-app-service

kubectl get svc my-app-service -o wide

curl http://<EXTERNAL-IP>

kubectl delete svc my-app-service

EXTERNAL NAME-ExternalName is a **special type of Kubernetes Service**.

Instead of routing traffic to Pods inside the cluster, it acts like a **DNS alias (CNAME record)**.

It maps the Service name to an **external DNS name** (e.g., google.com, mydb.example.com).

No selector, no endpoints, no pods needed!

External name-

apiVersion: v1

kind: Service

metadata:

name: ipl-svc

namespace: sinchan

spec:

type: ExternalName

externalName: flipkart.com # only one DNS allowed

ports:

- port: 80

Commands-

vi external.yaml

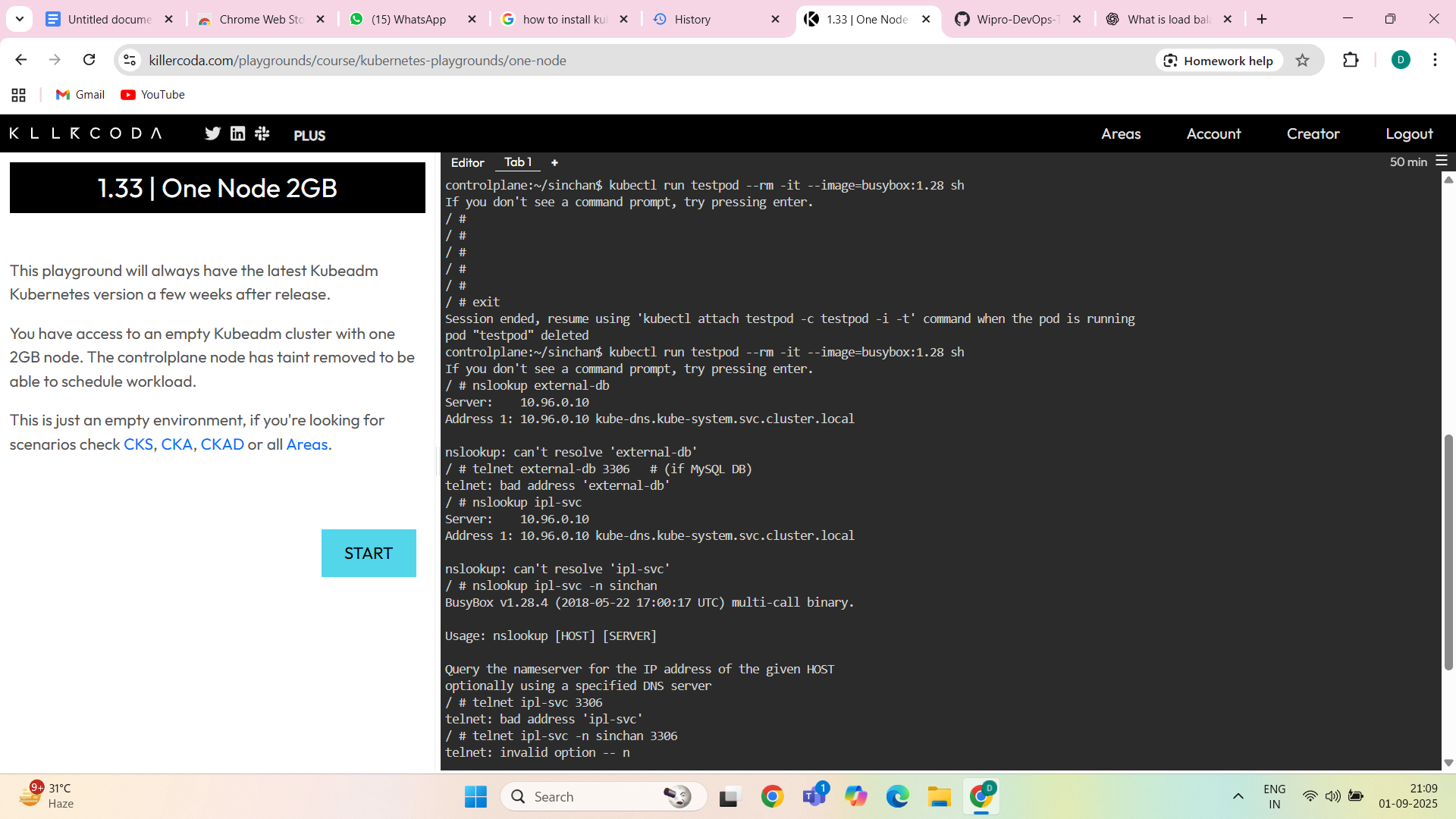
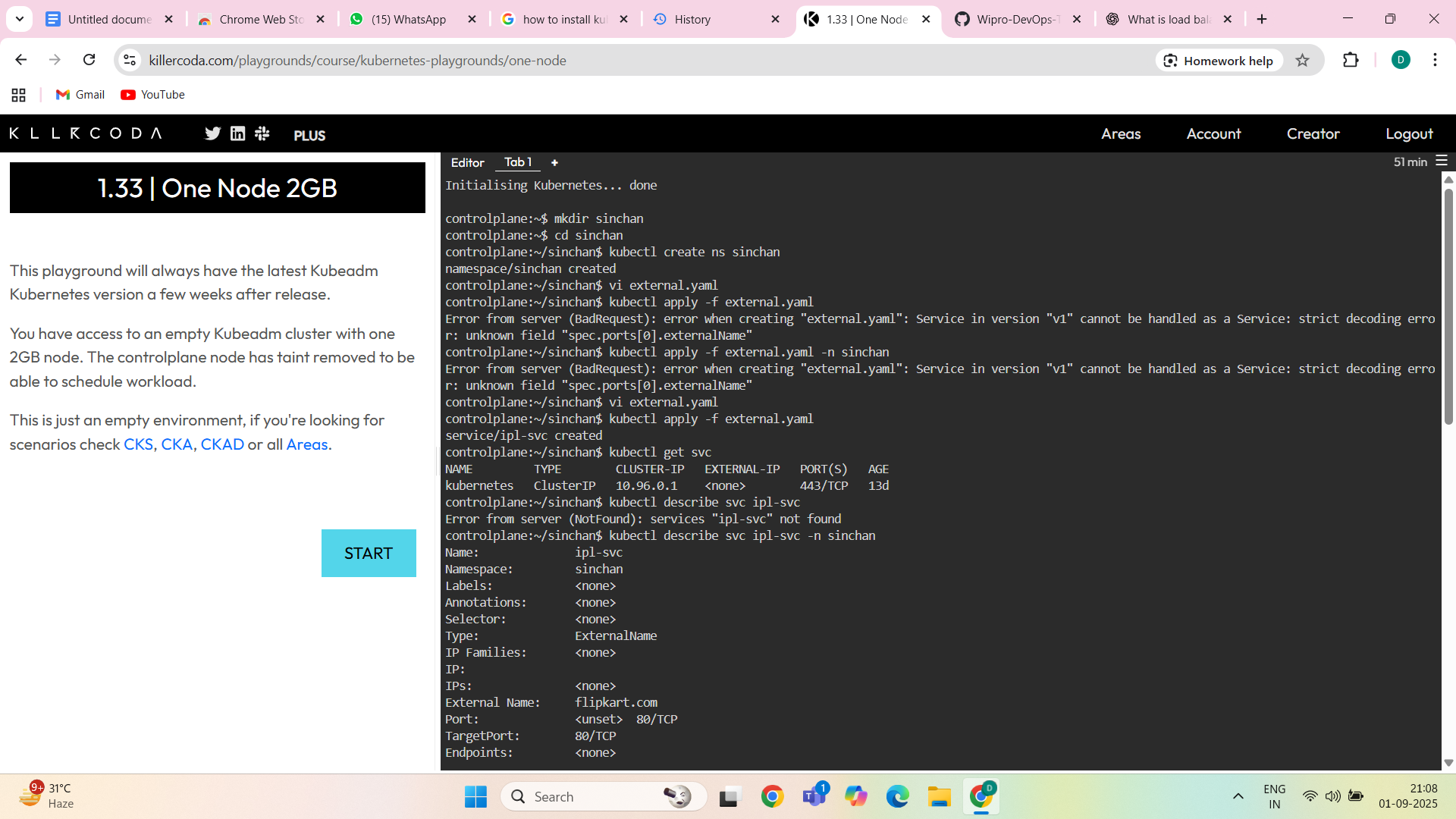
kubectl apply -f external.yaml

kubectl get svc

kubectl describe svc ipl-svc

kubectl describe svc ipl-svc -n sinchan

kubectl run testpod --rm -it --image=busybox:1.28 sh



Commands Explanation-

POD-

Podkubectl apply -f pod.yaml # Create Pod

kubectl get pods # List Pods

kubectl describe pod mypod # Pod details

kubectl logs mypod # See logs

kubectl exec -it mypod -- sh # Access container shell

kubectl delete pod mypod # Delete Pod

REPLICASET-

kubectl apply -f replicaset.yaml # Create RS

kubectl get rs # List ReplicaSets

kubectl scale rs myapp-rs --replicas=5 # Scale Pods

kubectl delete rs myapp-rs # Delete RS

DEPLOYMENT-

kubectl apply -f deployment.yaml # Create Deployment

kubectl get deployments # List Deployments

kubectl rollout status deployment myapp-deployment # Check rollout

kubectl rollout undo deployment myapp-deployment # Rollback

kubectl scale deployment myapp-deployment --replicas=5 # Scale

kubectl delete deployment myapp-deployment # Delete

-