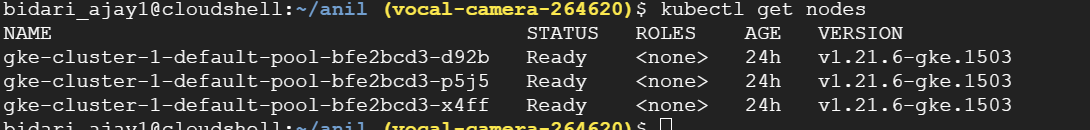
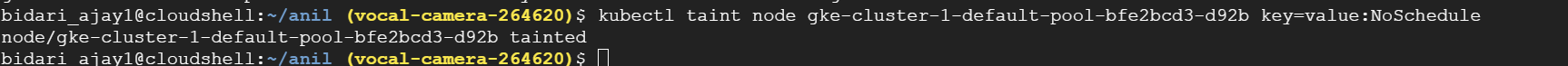
Taint and tolerations Nodes

Kubectl get nodes

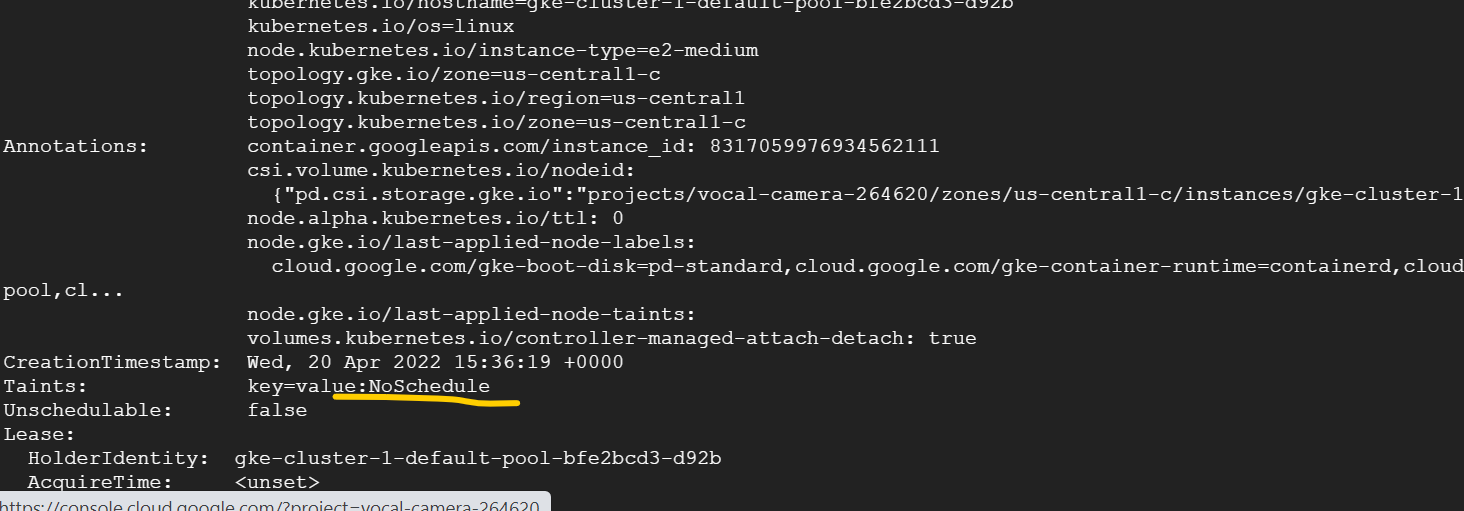


Taint first node in your node list

kubectl taint node gke-cluster-1-default-pool-58ed402f-23sn key=value:NoSchedule



kubectl describe node gke-cluster-1-default-pool-bfe2bcd3-d92b



vim deployment1.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: ajay

spec:

replicas: 3

selector:

matchLabels:

app: ajay

template:

metadata:

labels:

app: ajay

spec:

containers:

- name: ajay

image: nginx

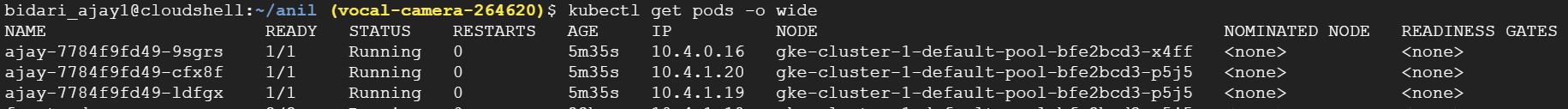
ports:

- containerPort: 80

kubectl create -f deployment1.yaml

Check no pod scheduled on the node which is tainted

kubectl get pods -o wide



vim deployment2.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: ajay2

spec:

replicas: 3

selector:

matchLabels:

app: ajay2

template:

metadata:

labels:

app: ajay2

spec:

containers:

- name: ajay2

image: nginx

ports:

- containerPort: 80

tolerations:

- key: "key"

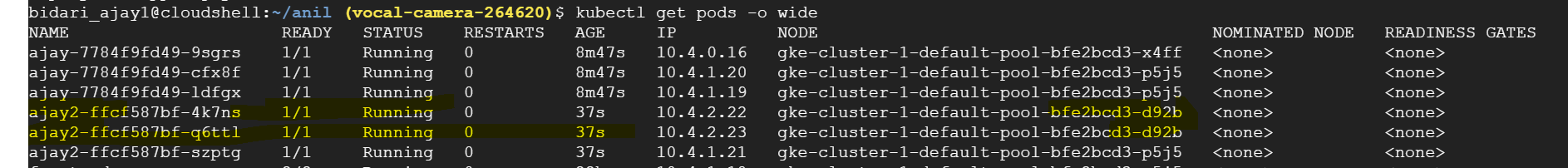
operator: "Exists"

effect: "NoSchedule"

kubectl create -f deployment2.yaml

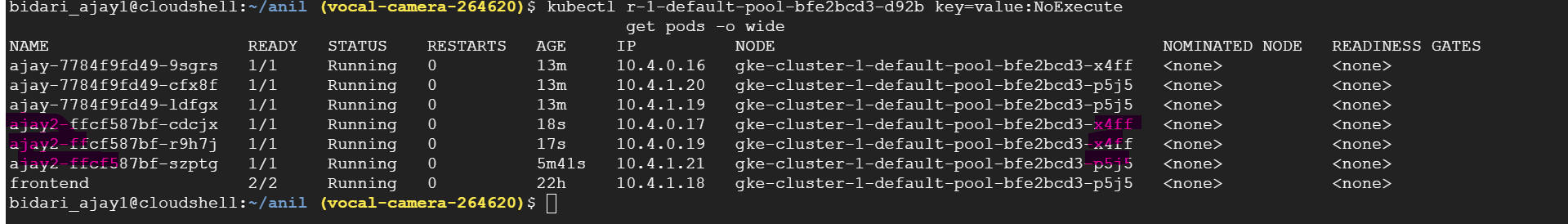
Now check pod gets deployed on tainted node as well because of tolerations for the pod to that node

kubectl get pods -o wide



kubectl taint node gke-cluster-1-default-pool-58ed402f-23sn key=value:NoExecute

taint other node with the value as NoExecute you should notice the node which is tainted with NoExecute does not have pods running now they are migrated to other nodes.



To untaint

kubectl taint node gke-cluster-1-default-pool-bfe2bcd3-d92b key=value:NoExecute-

kubectl taint node gke-cluster-1-default-pool-bfe2bcd3-d92b key=value:NoSchedule-