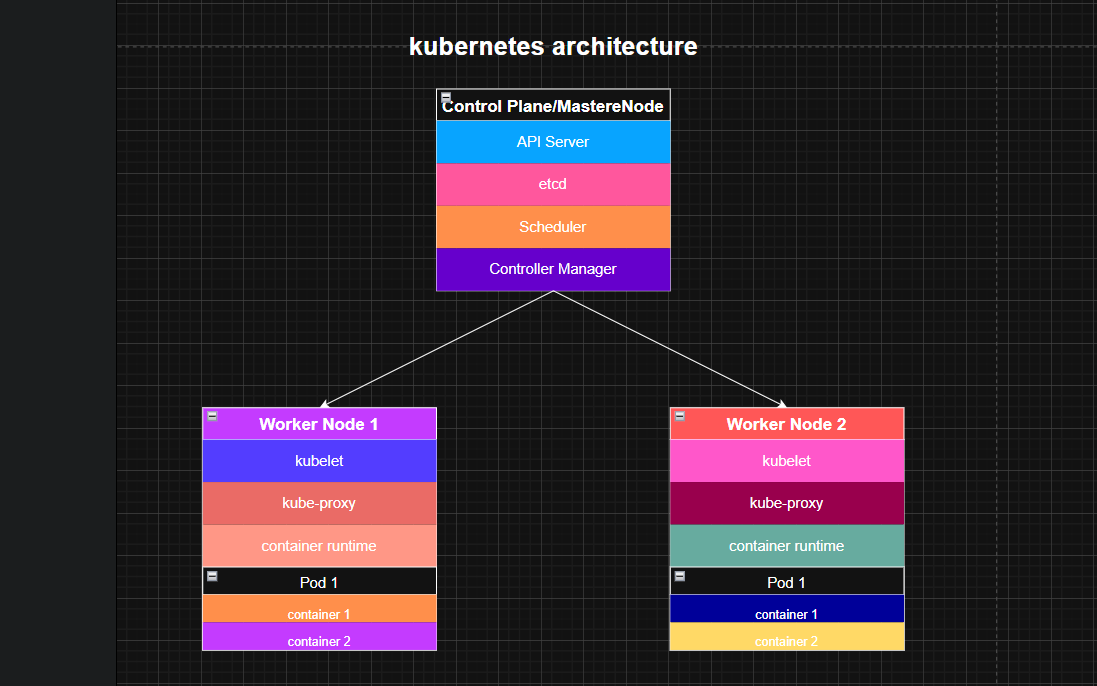
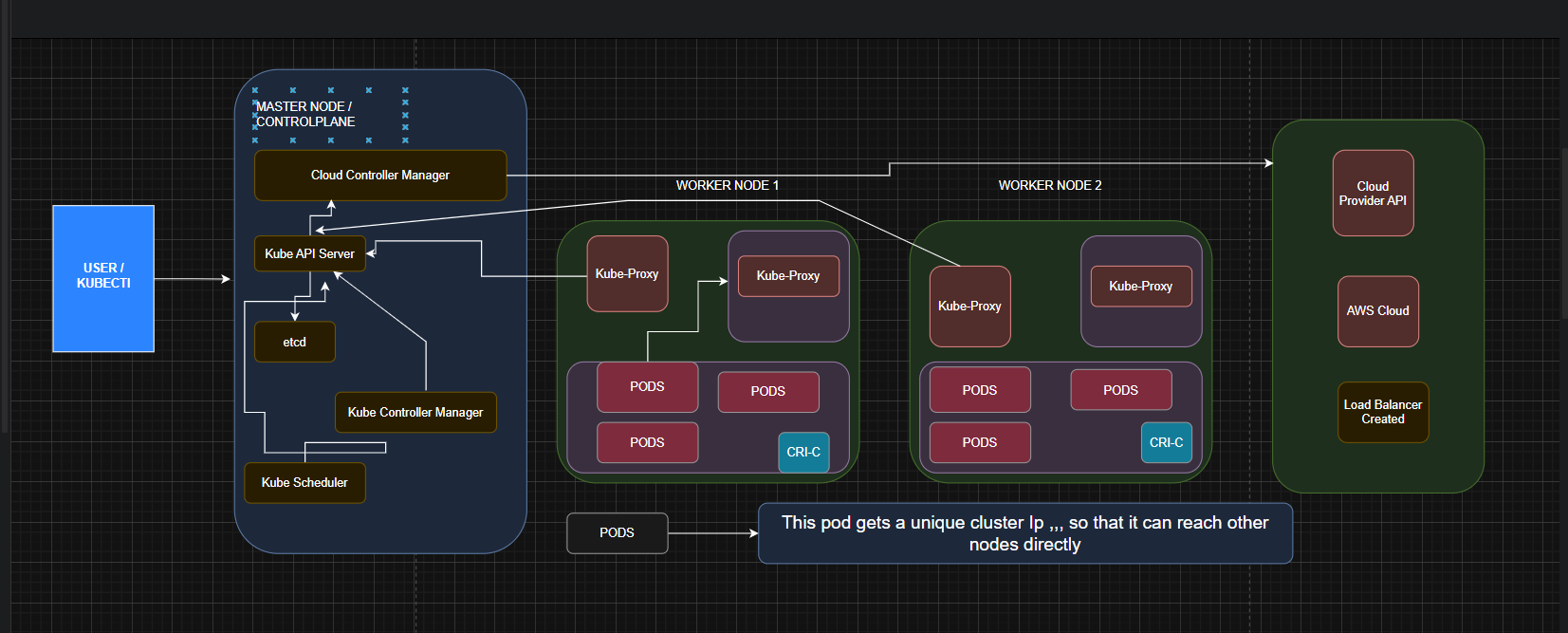
**TASK1**

**1. Draw the K8s architecture: -**



**Control Plane (Master)**: The brain of the cluster.

**API Server:** The front-end for the control plane; all communications go through it.

**etcd:** A consistent and highly-available key-value store for all cluster data.

**Scheduler:** Watches for newly created Pods and assigns them to a node.

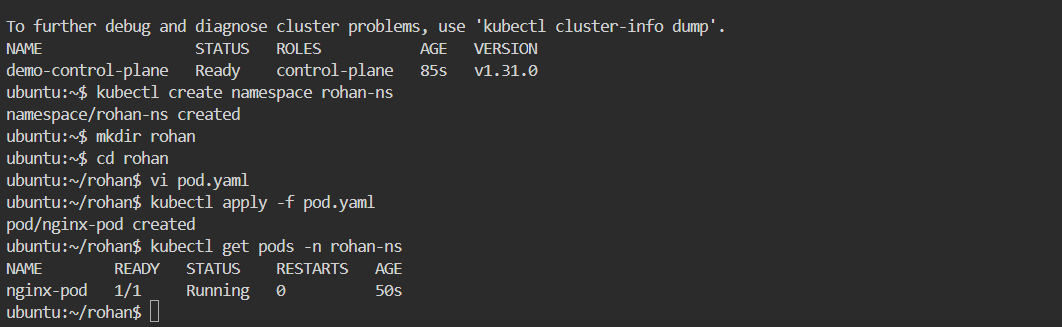
**Controller Manager:** Runs controller processes (e.g., Node Controller, Replication Controller).

**Worker Nodes:** The machines where your applications (workloads) run.

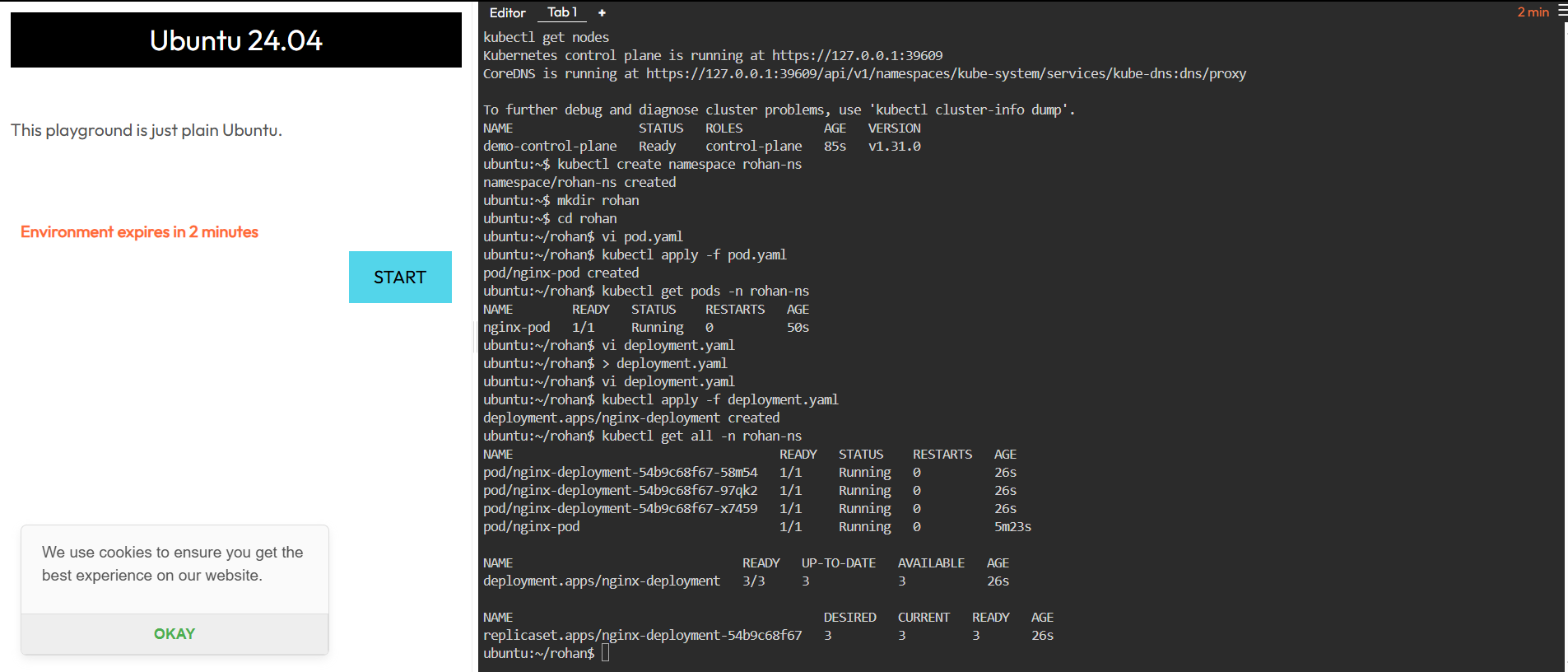
**Kubelet:** An agent that ensures containers are running in a Pod.

**Kube-proxy**: Maintains network rules to allow communication to your Pods.

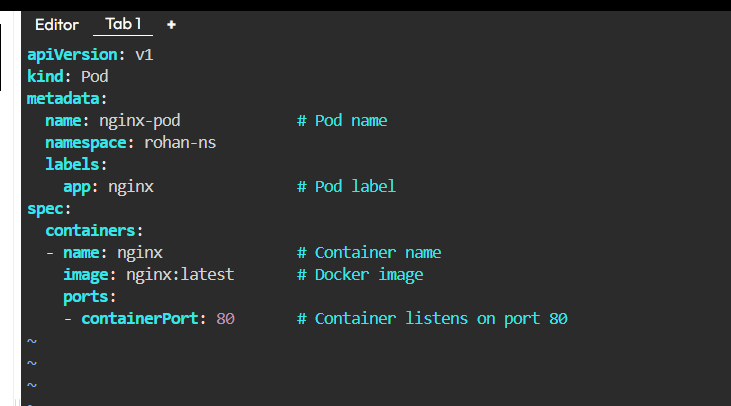
**Container Runtime:** The software that runs containers (e.g., containerd, CRI-O).

**TASK2**

| **Step 1:** Create a directory named rohan  mkdir rohan  **Step 2:** Go inside the directory  cd rohan  **Step 3:** Create a file named pod.yaml  vi pod.yaml  **Step 4:** Add Pod definition inside pod.yaml  Paste the following YAML content:  apiVersion: v1  kind: Pod  metadata:  name: nginx-pod  namespace: rohan-ns  labels:  app: nginx  spec:  containers:  - name: nginx  image: nginx:latest  ports:  - containerPort: 80 | Save and exit (esc, shift+:, wq).  **Step 5:** Apply the Pod in Kubernetes  Since I am inside the rohan directory:  kubectl apply -f pod.yaml  This will create a Pod named nginx-pod inside my namespace, rohan-ns. |
| --- | --- |



**pod.yaml**



define each n every instruction of pod yaml

### **apiVersion: v1**

* Tells Kubernetes which **API version** to use for this object.
* Pods belong to the **core group**, so we use v1.

### **kind: Pod**

* Defines the **type of resource** we are creating.
* Here it’s a **Pod** (the smallest deployable unit in Kubernetes).

### **metadata:**

This section contains **identification info** about the Pod.

* name: nginx-pod  
   → The Pod’s unique name in namespace rohan-ns.
* namespace: rohan-ns  
   → Puts this Pod inside your custom namespace (rohan-ns).
* labels:  
   → Key/value pairs to categorize the Pod.  
   → Example: app: nginx → helps Services/Deployments select this Pod.

### **spec:**

This is the **specification of how the Pod should run**.

* containers:  
   → A Pod can have one or multiple containers.  
   → Here we define only **one container**.

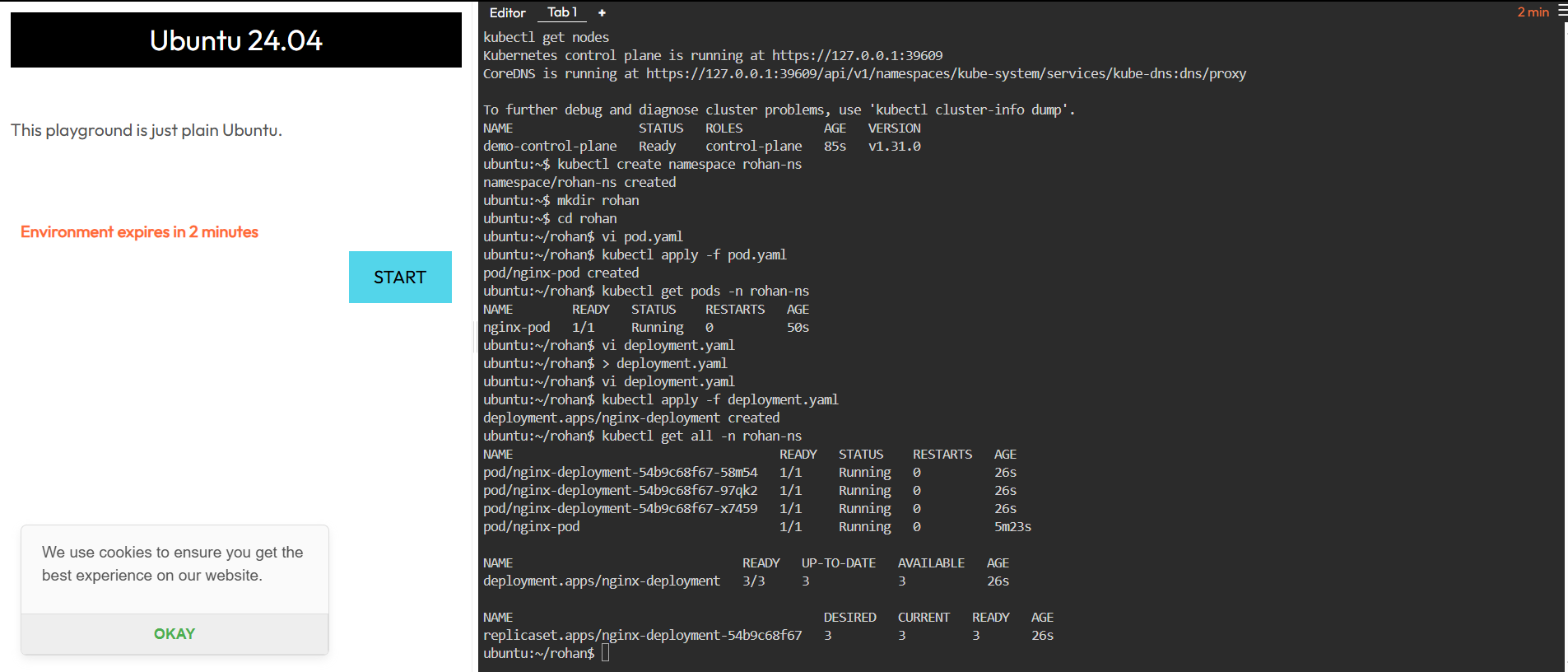
### **Inside containers:**

* name: nginx  
   → The container’s name inside the Pod.  
   → (Doesn’t need to match the Pod name, but usually related).
* image: nginx:latest  
   → The **Docker image** to run inside the container.  
   → In this case, it pulls the **official Nginx image** from Docker Hub.
* ports:  
   → Which ports inside the container should be exposed.  
  + containerPort: 80  
     → Tells Kubernetes that this container listens on **port 80** (the default HTTP port).  
     → This doesn’t expose it outside the cluster — it only marks port 80 as in use by the container (so Services can route to it).

**Task3**

**prepare all commands of pod in k8s**

**Create a Pod**

**kubectl apply -f pod.yaml**

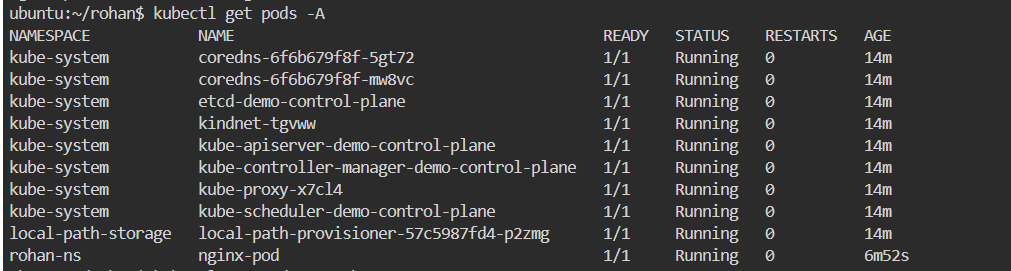
**List Pods**

**kubectl get pods**

****

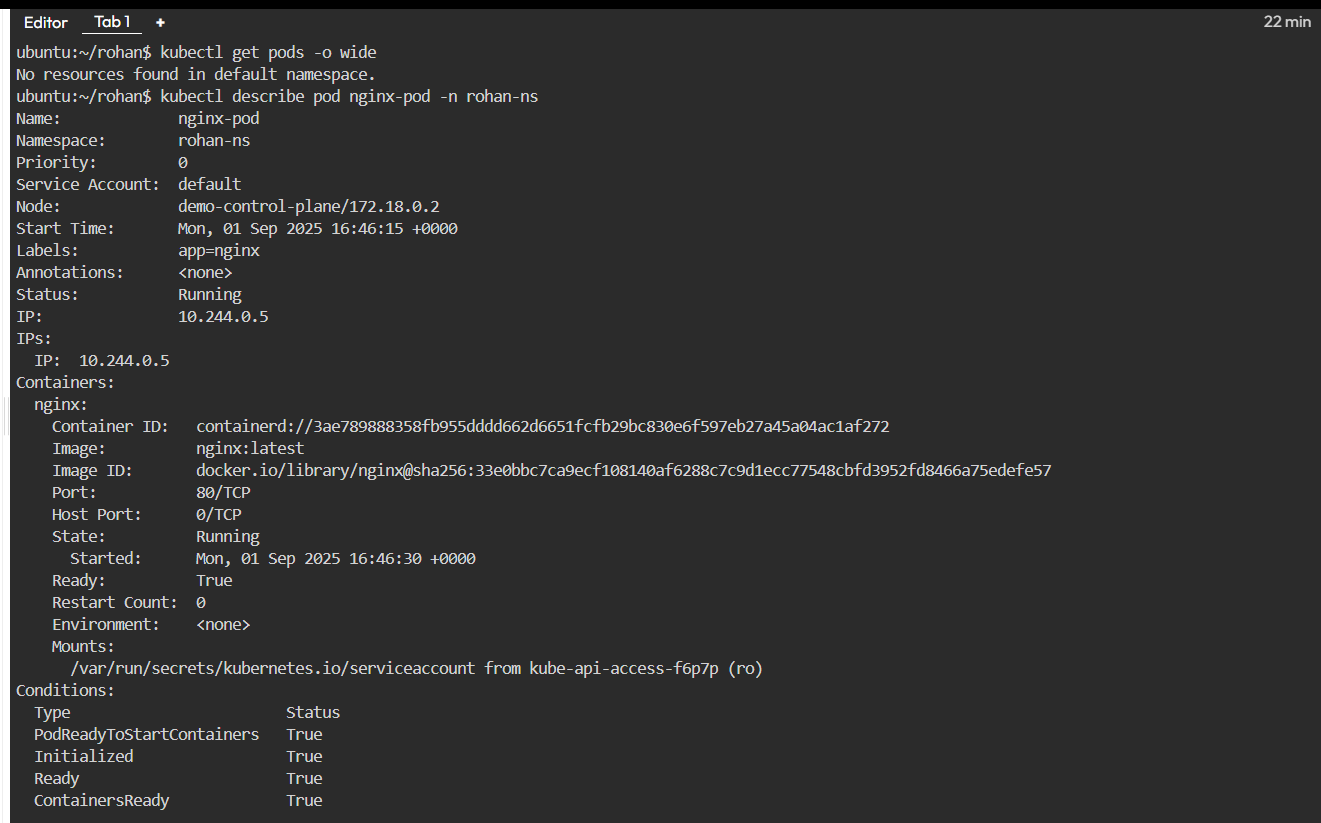
**In all namespaces:**

**kubectl get pods -A**

****

**In specific namespace:**

**kubectl get pods -n rohan-ns**

****

**With wide output (Node, IP, etc.):**

**kubectl get pods -o wide**

**Get Pod Details**

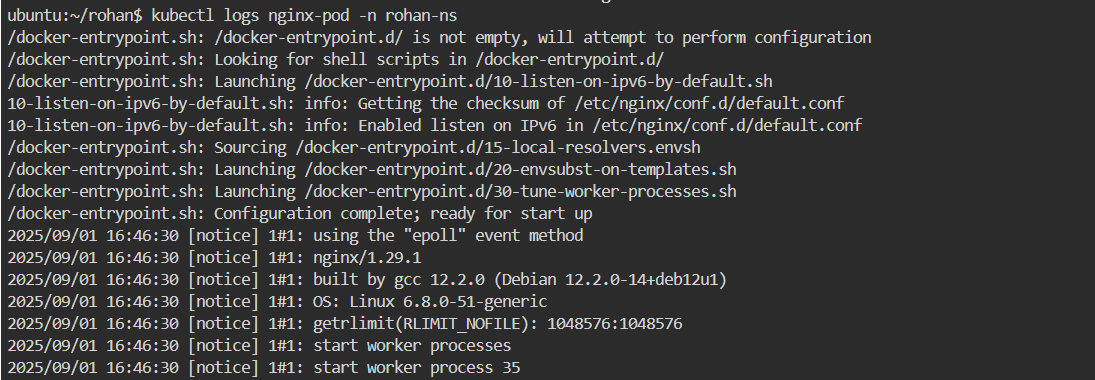
**Describe a Pod:**

**kubectl describe pod nginx-pod -n rohan-ns**

**Logs from Pod**

**View logs:**

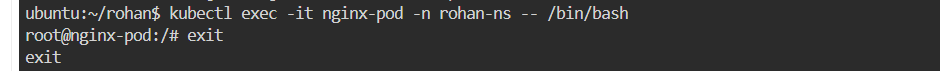
**kubectl logs nginx-pod -n rohan-ns**

****

**Access a Pod**

**Open a shell inside container:**

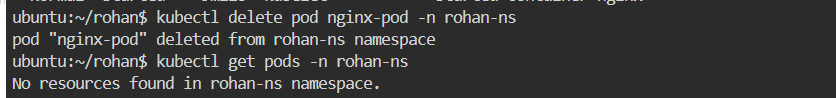
**kubectl exec -it nginx-pod -n rohan-ns -- /bin/bash**

****

**Delete Pod**

**By name:**

**kubectl delete pod nginx-pod -n rohan-ns**

****

**Check Pod status continuously:**

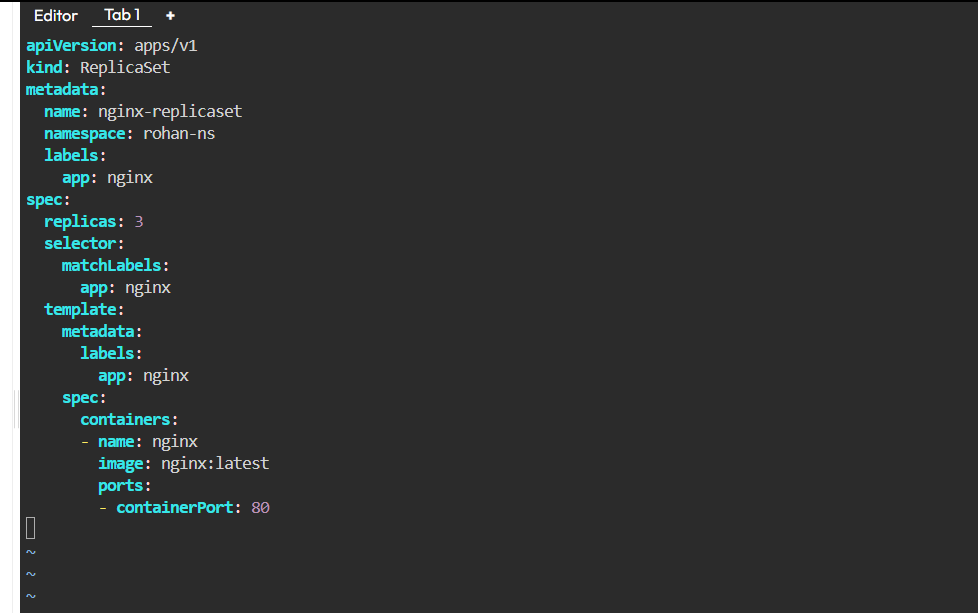
**kubectl get pods -n rohan-ns -w**

**Restart Pod (delete → re-create):**

**kubectl delete pod nginx-pod -n rohan-ns**

**kubectl apply -f pod.yaml**

**TASK-4**



### 

### 

### 

### **apiVersion:**

* Tells Kubernetes which API group/version this resource belongs to.
* apps/v1 is the correct API for ReplicaSets, Deployments, StatefulSets, etc.

### **kind: ReplicaSet**

* Defines the **resource type** → here we are creating a **ReplicaSet**.
* ReplicaSet ensures a **specified number of Pods** are always running

### **metadata:**

Information to **identify** the ReplicaSet.

* name: nginx-replicaset → Name of this ReplicaSet (unique in namespace).
* namespace: rohan-ns → This ReplicaSet lives in the namespace rohan-ns.
* labels: → Key/value tags to categorize this object.  
  app: nginx → Useful for grouping and selecting resources.

### **spec:**

This section tells Kubernetes **how the ReplicaSet should behave**.

#### **replicas: 3**

* Desired **number of Pods** to run.
* ReplicaSet will make sure **3 Pods are always running**

#### **selector:**

* Defines **which Pods** this ReplicaSet is responsible for managing.
* matchLabels: → Pods must have the label app: nginx to be controlled by this ReplicaSet.
* **Important**: If labels in selector do not match labels in template.metadata.labels, ReplicaSet will never create Pods.

#### **template:**

This is the **Pod template** (like pod.yaml) that ReplicaSet will use to create Pods.

* metadata:  
  + labels: → Pods created will get label app: nginx.
  + These must match the selector.
* spec:  
  + Defines containers inside each Pod.

#### **Inside spec.containers:**

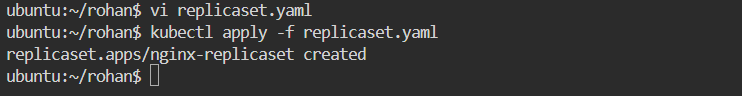
* - name: nginx  
   → Name of the container in the Pod.
* image: nginx:latest  
   → The Docker image to use.  
   → Kubernetes will pull nginx:latest from Docker Hub.
* ports:  
  + containerPort: 80 → Container listens on port **80** (HTTP).
  + This doesn’t expose it outside the cluster, but tells K8s what port this container uses.

**TASK-5**

**prepare all commands of rs in k8s**

Create a ReplicaSet

kubectl apply -f replicaset.yaml

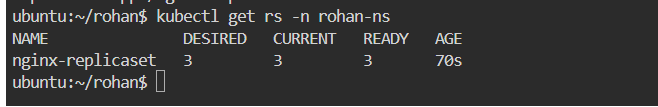


List ReplicaSets

In current namespace:

kubectl get rs

In a specific namespace:

kubectl get rs -n rohan-ns

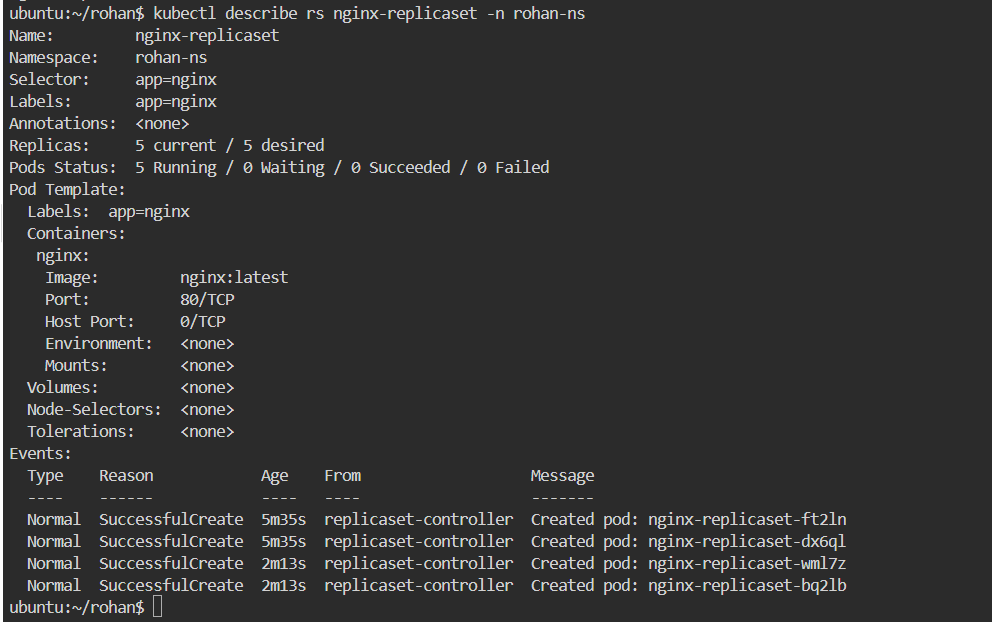
With more details (labels, age, etc.):

kubectl get rs -o wide

Get ReplicaSet Details

Describe ReplicaSet:

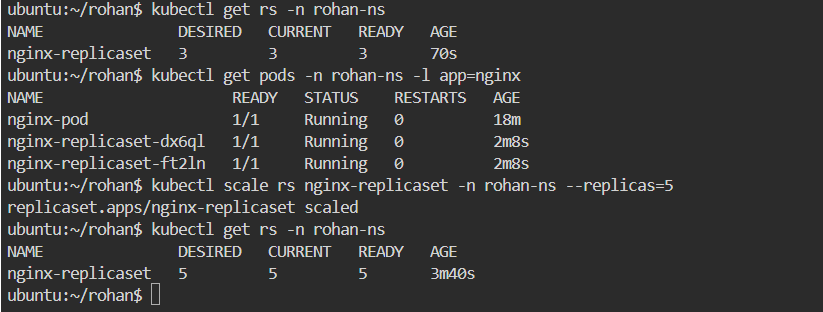
kubectl describe rs nginx-replicaset -n rohan-ns



Scale ReplicaSet

Scale replicas (e.g., 5 Pods):

kubectl scale rs nginx-replicaset --replicas=5 -n rohan-ns



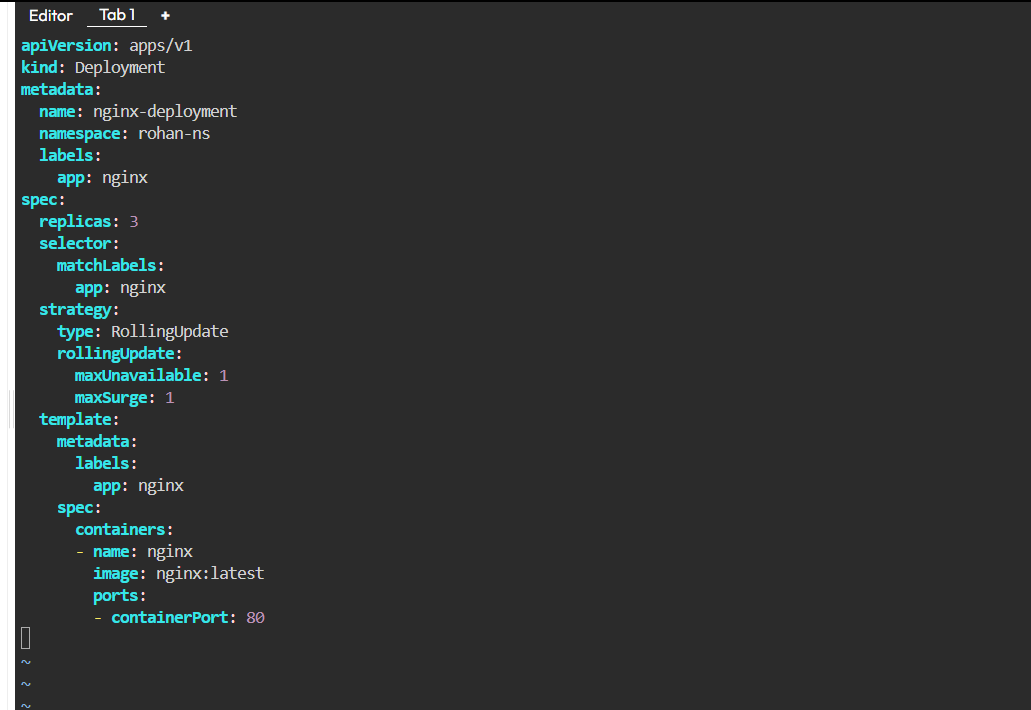
Delete ReplicaSet

By name:

kubectl delete rs nginx-replicaset -n rohan-ns

**TASK 6**

6.write deployment yaml , create it and define each n every instruction of deployment yam



l**apiVersion: apps/v1**

* Tells Kubernetes **which API version** to use.
* apps/v1 is correct for Deployments, ReplicaSets, StatefulSets, etc.

### **kind: Deployment**

* Specifies the **resource type** we are creating.
* Here it’s a **Deployment**, which manages **ReplicaSets** internally to handle rollouts and rollbacks.

### **metadata:**

General info to **identify** the Deployment.

* name: nginx-deployment → Name of Deployment (unique inside namespace).
* namespace: rohan-ns → This Deployment will be created in namespace rohan-ns.
* labels: → Key/value tags to organize and select objects.  
  + app: nginx → A label used later by Services, ReplicaSets, etc.

### **spec:**

This defines the **desired state** of the Deployment

#### **replicas: 3**

* The Deployment should ensure **3 Pods** are running at all times.
* If a Pod dies, a new one is created.

#### **selector:**

* Defines **which Pods belong to this Deployment**.
* matchLabels: → Only Pods with app: nginx will be managed.
* Important: Must match template.metadata.labels below.

#### **strategy:**

Defines the **update strategy** when Pods are updated.

* type: RollingUpdate → Default strategy. Updates Pods **gradually**, avoiding downtime.
* rollingUpdate: → Controls update behavior:  
  + maxUnavailable: 1 → At most **1 Pod** can be unavailable during update.
  + maxSurge: 1 → At most **1 extra Pod** can be created (surge capacity).

#### **template:**

This is the **Pod template** (like a pod.yaml) that Deployment uses to create Pods.

##### **metadata:**

* Labels applied to Pods created by this Deployment.
* app: nginx → Must match the selector.

##### **spec:**

Defines containers inside each Pod.

###### **containers:**

* A list of containers that run in the Pod.
* - name: nginx → Name of the container inside the Pod.
* image: nginx:latest → Docker image to pull (from Docker Hub by default).
* ports:  
  + containerPort: 80 → Exposes port 80 inside the Pod.  
     (Doesn’t expose externally; used by Services to connect to this Pod).

# **🔹 Summary of Flow**

1. **Deployment** manages **ReplicaSets** → ReplicaSets manage **Pods**.
2. Ensures **3 Pods** are always running (replicas: 3).
3. Uses **labels** to group and manage Pods.
4. Performs **rolling updates** with no downtime.
5. If something goes wrong, you can **rollback** easily.

**TASK7**

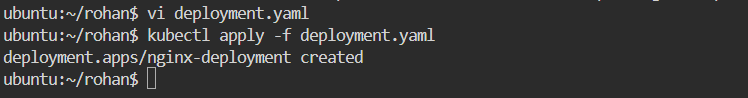
**prepare all commands of deployment in k8s:-**

# 1. Create namespace

kubectl create namespace rohan-ns

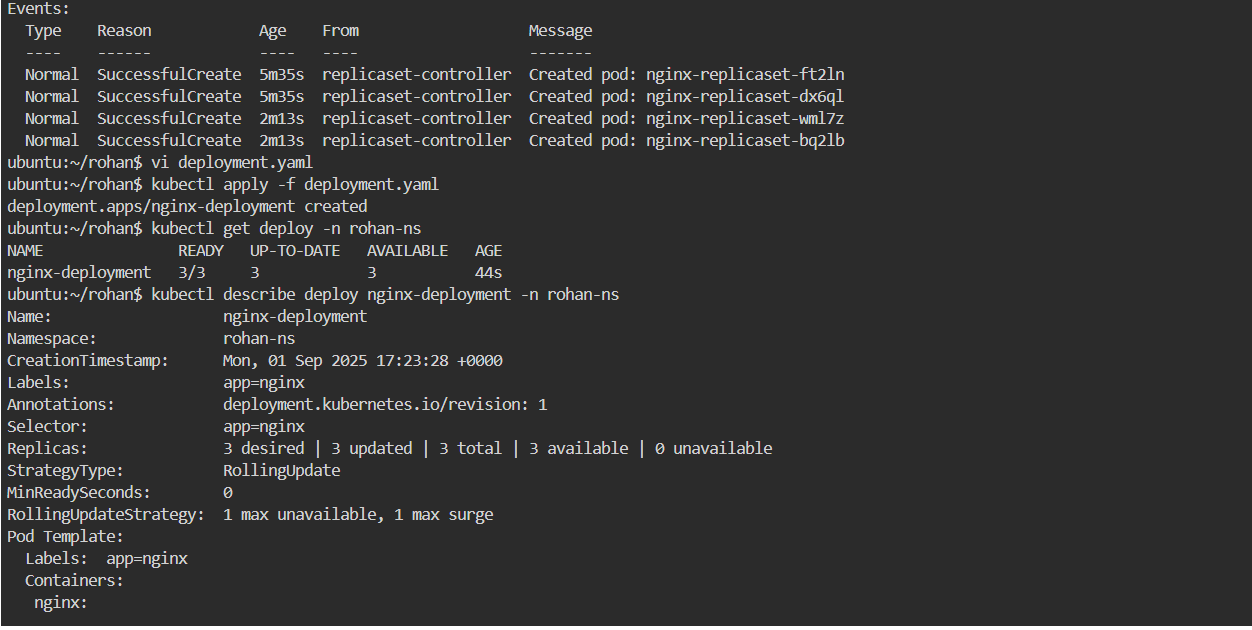
# 2. Create Deployment

kubectl apply -f deployment.yaml



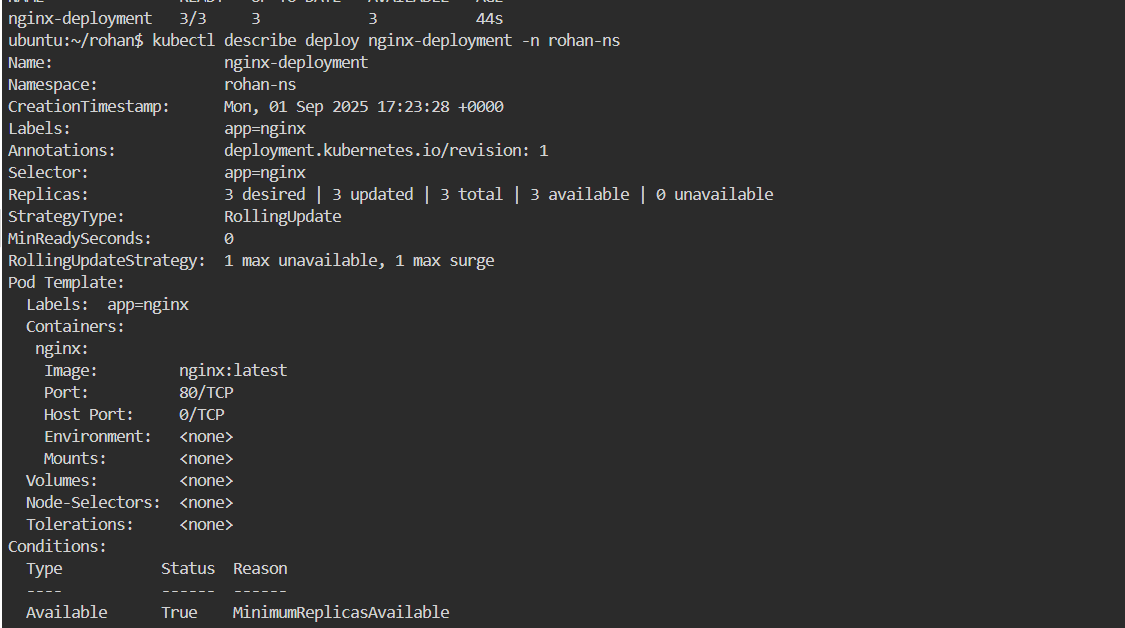
# 3. List Deployments

kubectl get deploy -n rohan-ns



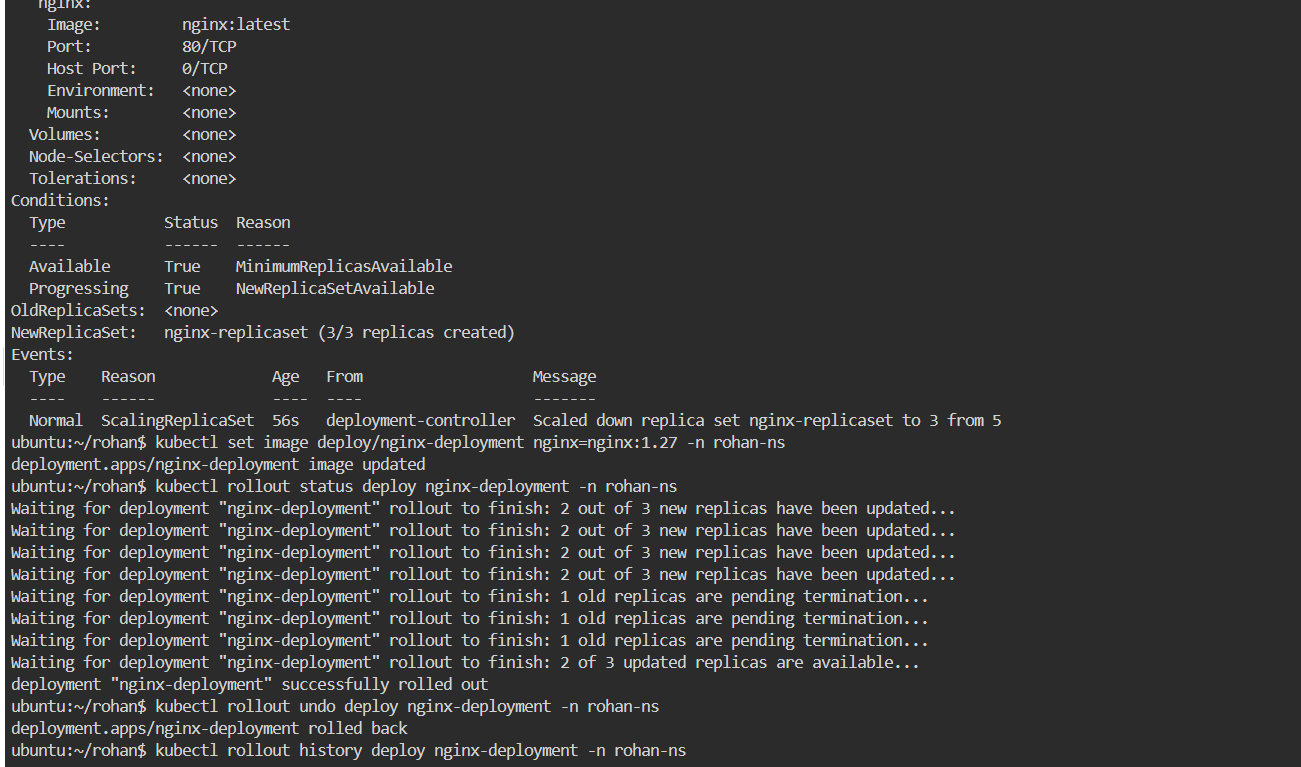
# 4. Describe Deployment

kubectl describe deploy nginx-deployment -n rohan-ns



# 5. Scale Deployment

kubectl scale deploy nginx-deployment --replicas=5 -n rohan-ns



# 6. Update image

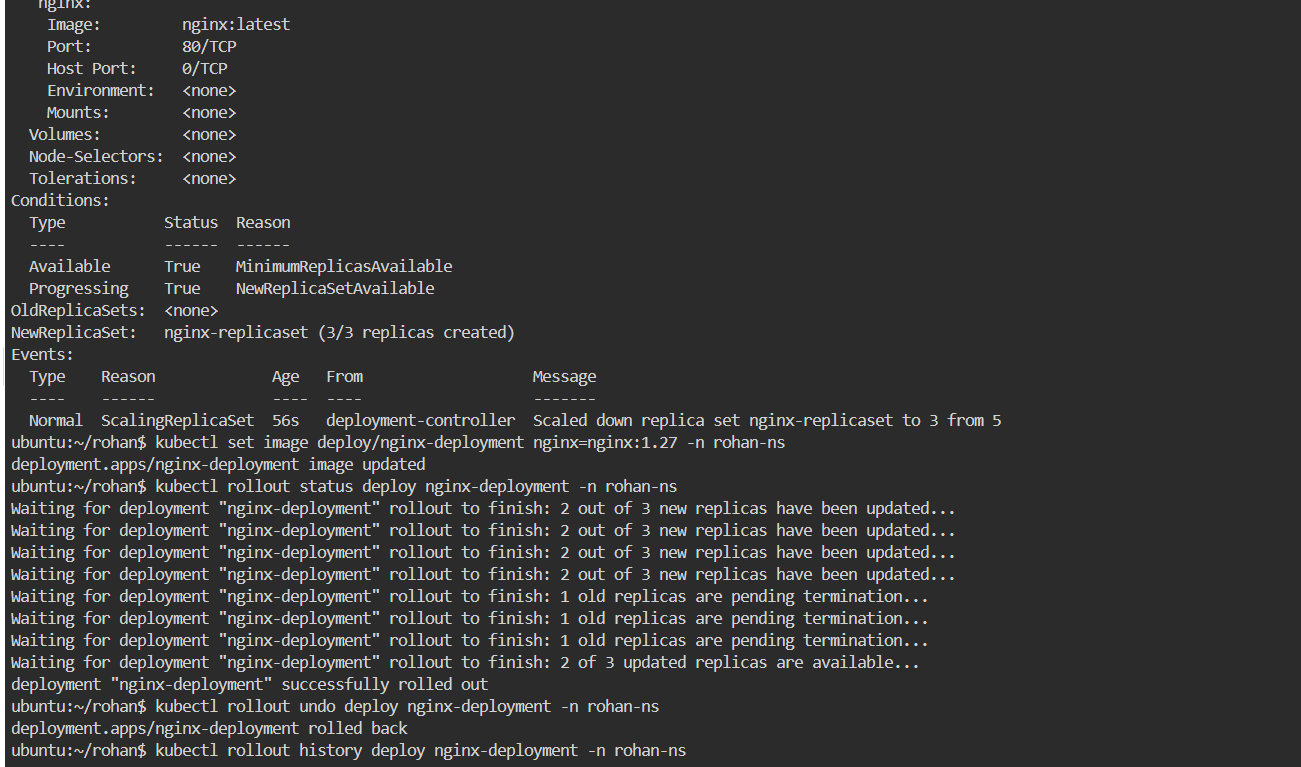
kubectl set image deploy/nginx-deployment nginx=nginx:1.25-alpine -n rohan-ns

# 7. Monitor rollout

kubectl rollout status deploy/nginx-deployment -n rohan-ns

# 8. Rollback

kubectl rollout undo deploy/nginx-deployment -n rohan-ns



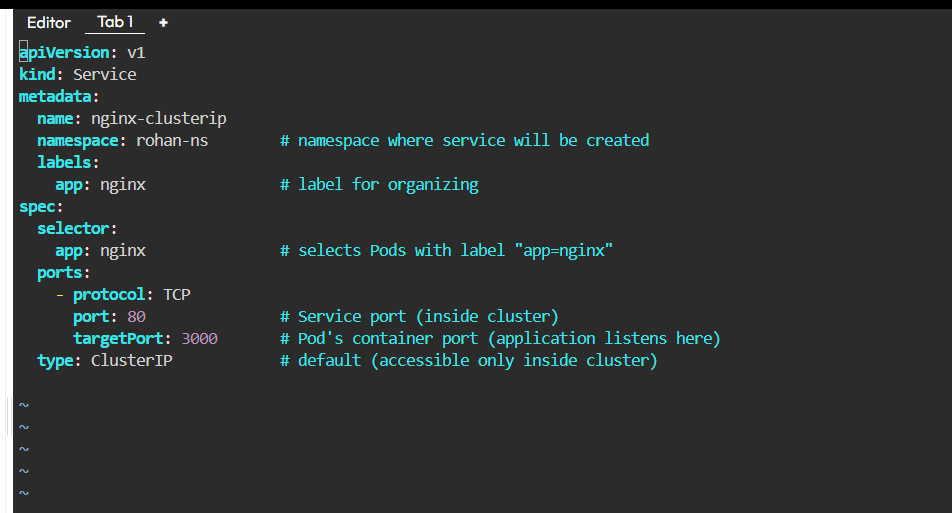
# 9. Delete Deployment

kubectl delete deploy nginx-deployment -n rohan-ns

**TASK-8**

**write service yaml file for all 4 types and define each n every instruction of svc yaml :-**

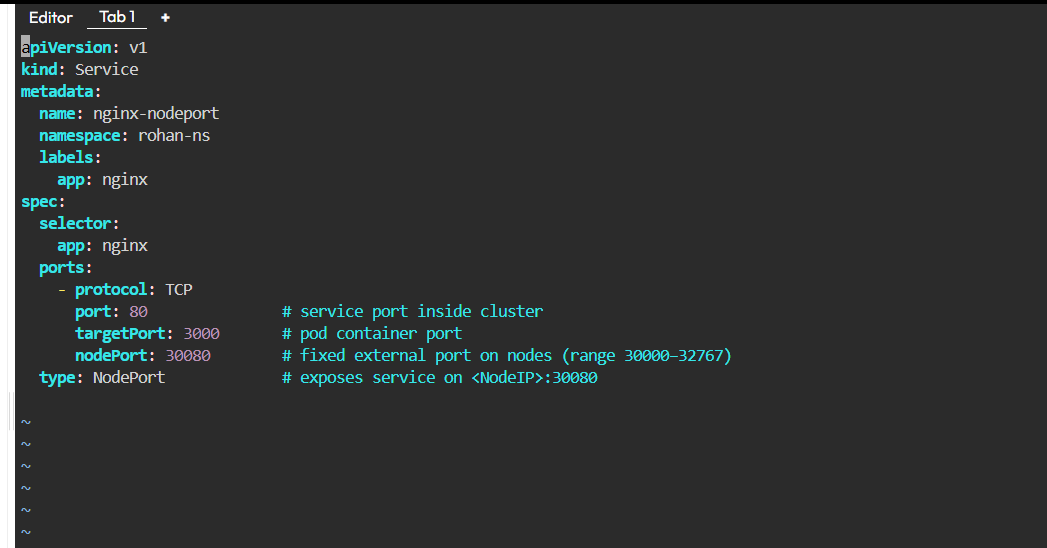
**1) CLUSTER\_IP:-**



### **Explanation**

* apiVersion: v1 → Core API group for Services.
* kind: Service → Defines a Service resource.
* metadata:  
  + name: nginx-clusterip → Unique name of the Service.
  + namespace: rohan-ns → Created in namespace rohan-ns.
  + labels: → Tag for categorization.
* spec: → Defines Service behavior.  
  + selector: → Matches Pods with app: nginx label.
  + ports: → Defines how traffic is forwarded.  
    - protocol: TCP → Protocol used.
    - port: 80 → Port exposed by the Service (inside cluster).
    - targetPort: 80 → Port on the Pod container.
  + type: ClusterIP → Accessible only inside the cluster (default type).

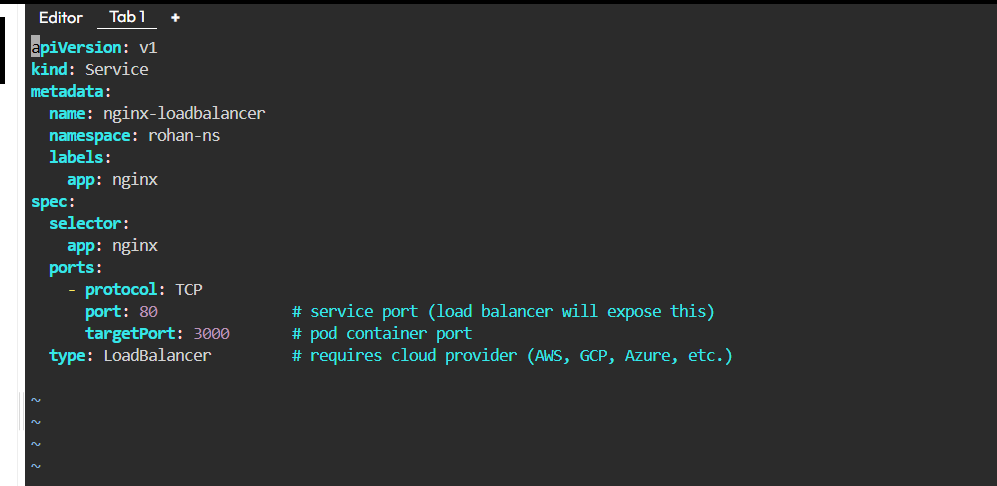
**2) NODE\_PORT :-**



### **Explanation**

* type: NodePort → Exposes Service externally via Node’s IP.
* nodePort: 30080 → Fixed port on each Worker Node (range: 30000–32767).

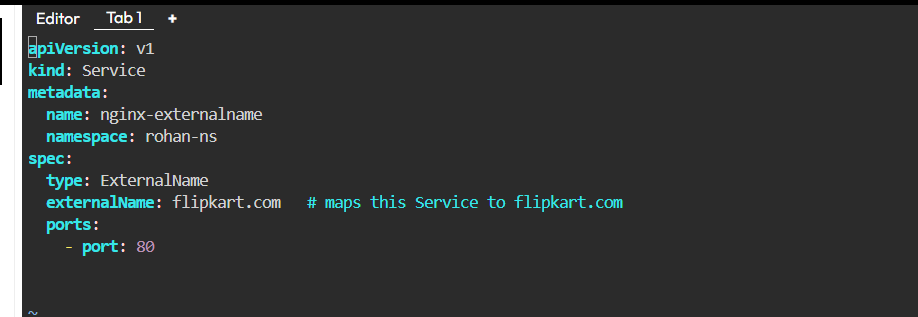
**3) Load\_Balancer :-**



### **Explanation**

* type: LoadBalancer → Creates a **Cloud Load Balancer** (works in AWS, GCP, Azure, etc.).
* port: 80 → External LB listens on port 80.
* targetPort: 80 → Traffic forwarded to Pods’ port 80.
* Provides a public IP assigned by cloud provider.

**4) EXTERNAL\_NAME\_SERVICE:-**



### **Explanation**

* type: ExternalName → No selector, no Pods, no ports.
* externalName: www.google.com → Maps this Service name (nginx-externalname) to an external DNS name.

**TASK=9**

**prepare all commands of service in k8s:-**

Create a Service

**kubectl apply -f service.yaml**

List Services

In current namespace:

**kubectl get svc**

In a specific namespace:

**kubectl get svc -n rohan-ns**

With more details:

**kubectl get svc -o wide -n rohan-ns**

Describe Service

**kubectl describe svc nginx-clusterip -n rohan-ns**

Delete Service

By name:

**kubectl delete svc nginx-clusterip -n rohan-ns**

