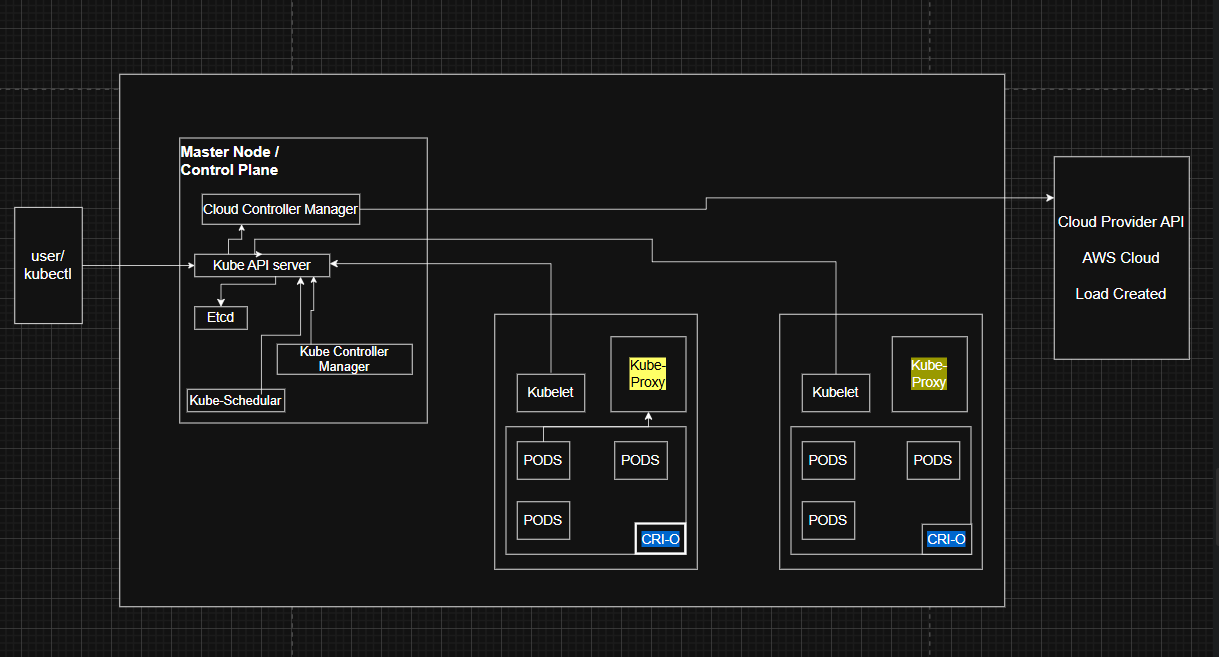
# **K8s Resources**

**K8s architecture**



**Pod:**

A Pod in Kubernetes is the smallest deployable unit that encapsulates one or more containers, giving them a shared network, storage, and configuration, and is used to run application workloads on a single node.

**Pod.yaml**

apiVersion: v1

kind: Pod

metadata:

  name: sunmarsh-pod

  namespace: sunmarsh

  labels:

    app: sunmarsh-nginx

spec:

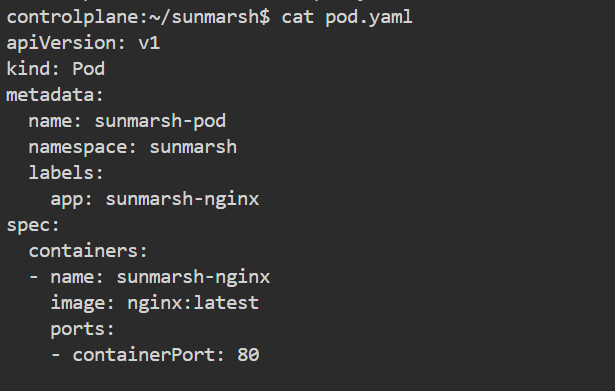
  containers:

  - name: sunmarsh-nginx

    image: nginx:latest

    ports:

    - containerPort: 80



**apiVersion: v1**

* Every Kubernetes object has an **API version** that tells the cluster which version of the Kubernetes API to use.
* v1 means this object belongs to the **core API group** (Pods are part of the core group).

**kind: Pod**

* Defines the **type of Kubernetes object** to create.
* Here, it is a **Pod** (the smallest deployable unit).

**metadata:**

* Section that holds basic information about the object (name, namespace, labels, annotations, etc.).

**name: sunmarsh-pod**

* The name of the Pod, unique within its namespace.
* This name will be used when you run commands like kubectl get pod sunmarsh-pod

**namespace: sunmarsh**

* The Pod will be created inside the sunmarsh namespace (not the default one).
* Namespaces help in logically separating resources (like dev, test, prod).

**labels:**

**app: sunmarsh-nginx**

* Labels are key–value pairs attached to objects.
* Here, app: sunmarsh-nginx can be used by a Service or Deployment selector to target this Pod.
* Labels also help in grouping and filtering resources.

**spec:**

* Defines the desired state or specification of the Pod.
* Everything under spec tells Kubernetes what to run inside this Pod.

**containers:**

* A Pod can have one or more containers.
* This section defines the list of containers that will run inside the Pod.

**- name: sunmarsh-nginx**

* The name of the container inside the Pod.
* Must be unique within the Pod (used for logs, exec, etc.).

**image: nginx:latest**

* The container image to run, here it’s nginx with the latest tag.
* This will be pulled from the default container registry (Docker Hub, unless configured otherwise).

**ports:**

* Declares which ports the container will listen on.
* This is mostly for documentation and Service discovery; it doesn’t expose the port outside by itself.

**- containerPort: 80**

* Defines that the container listens on port 80 (default HTTP port).
* Useful when a Service needs to target this Pod.

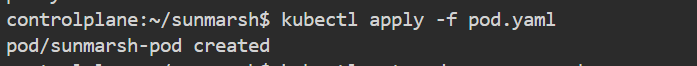
**Pod commands**

**1.Create a Pod**

kubectl apply -f pod.yaml

* Creates a Pod defined in the YAML file (pod.yaml).
* Pods are the smallest deployable units in Kubernetes.

**kubectl apply -f pod.yaml**

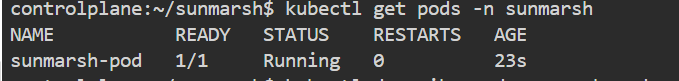
****

**2.View Pods**

kubectl get pods -n <namespace>

* Lists all Pods in the cluster or in a specific namespace.
* Displays NAME, READY (containers ready), STATUS (Running, Pending, etc.), RESTARTS, and AGE.

**kubectl get pods -n Sunmarsh**

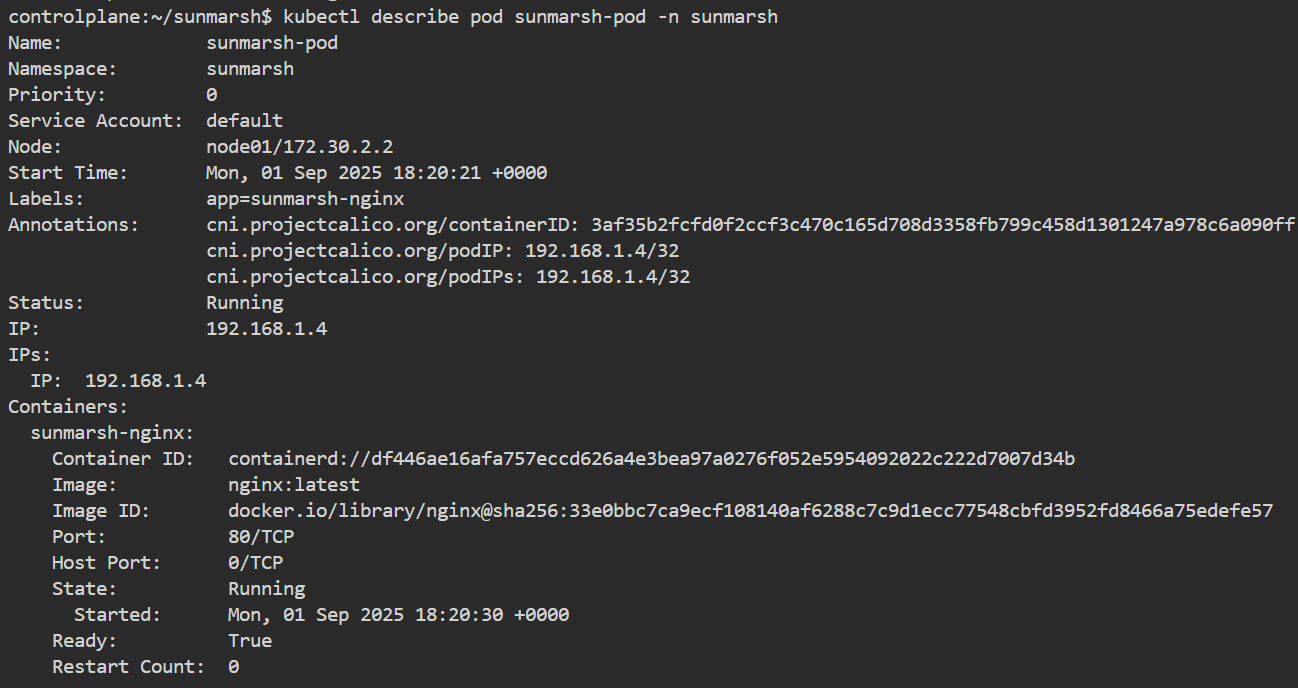
****

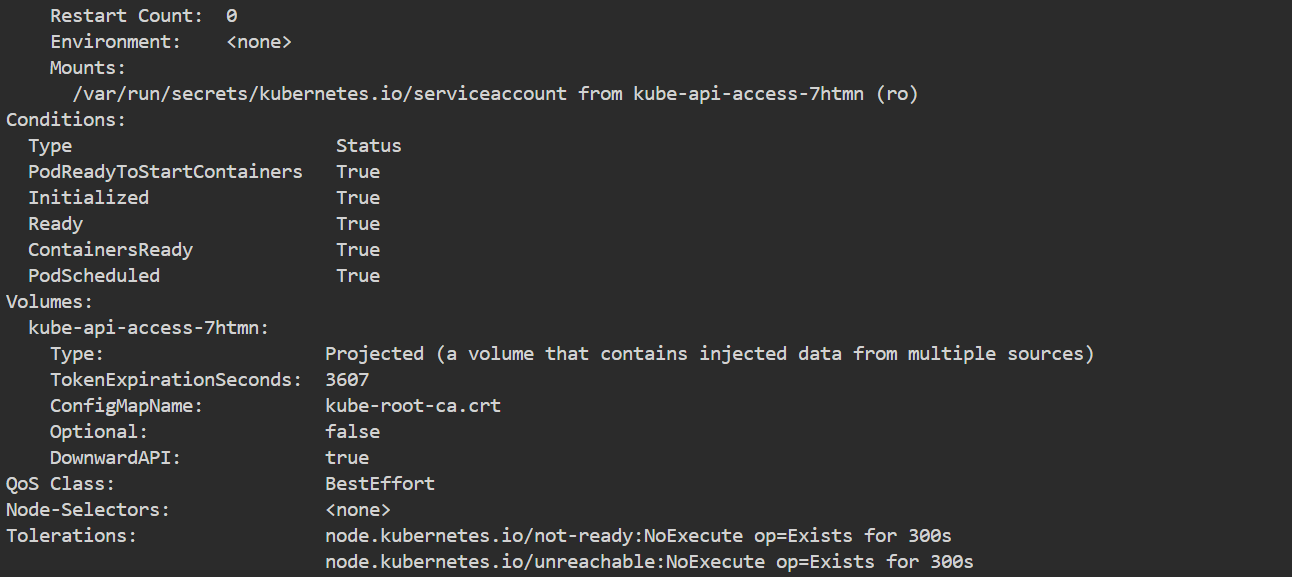
**3. Describe a Pod**

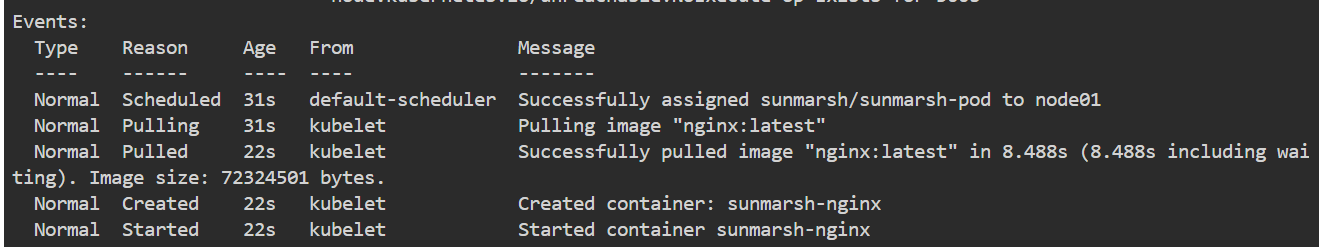
kubectl describe pod <pod-name> -n <namespace>

* Provides detailed information about a Pod:
  + Node it’s running on
  + Containers and their ports
  + Labels and annotations
  + Events (like crashes or restarts)

**kubectl describe pod sunmarsh-pod -n Sunmarsh**

****

****

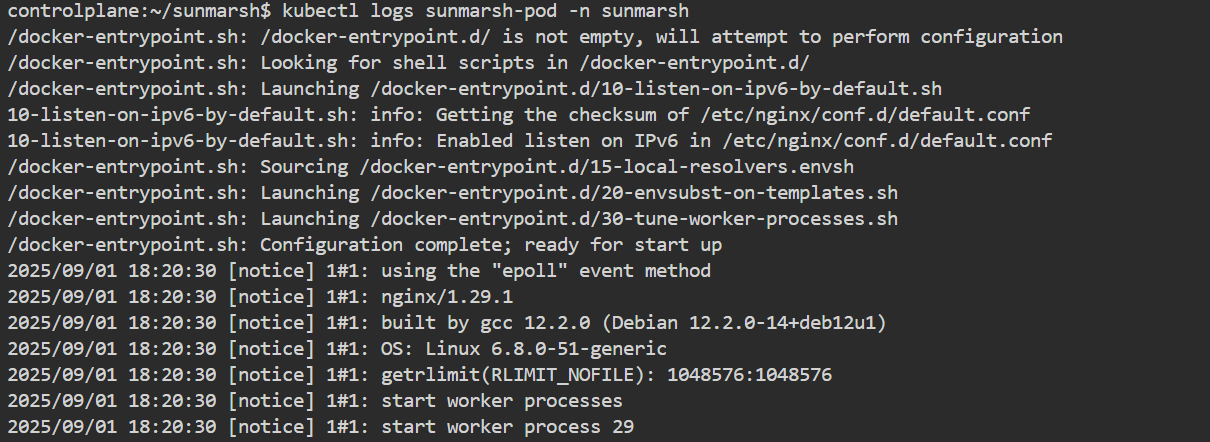
****

**4.Get Pod Logs**

kubectl logs <pod-name> -n <namespace>

* Shows the logs of a Pod’s container(s).

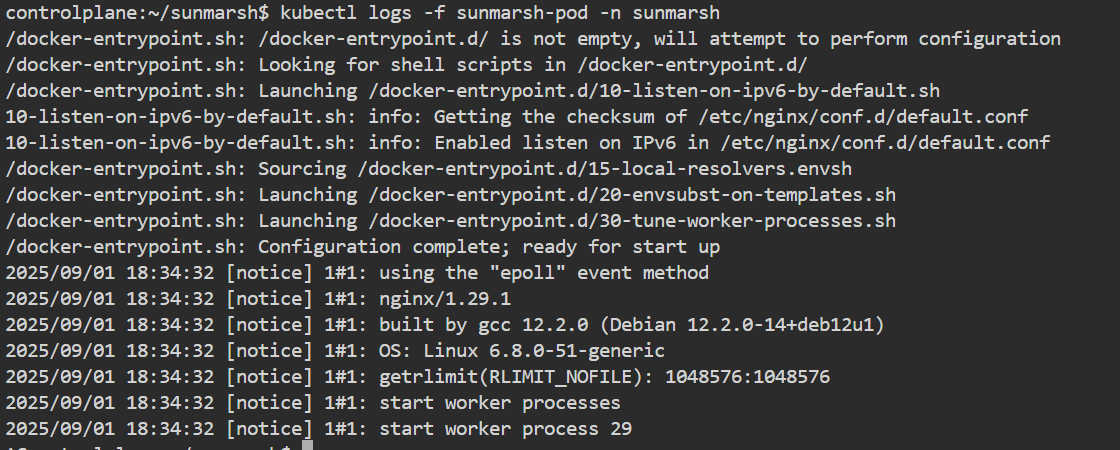
**kubectl logs sunmarsh-pod -n Sunmarsh**

****

* Add -f for real-time log streaming:

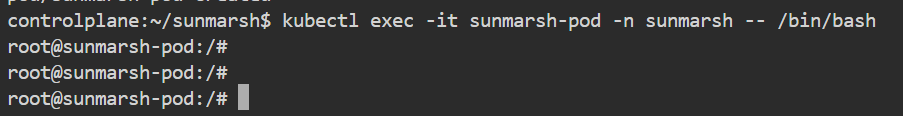
kubectl logs -f <pod-name> -n <namespace>

**kubectl logs -f sunmarsh-pod -n Sunmarsh**

****

**Execute Commands in a Pod**

**kubectl exec -it sunmarsh-pod -n sunmarsh -- /bin/bash**

****

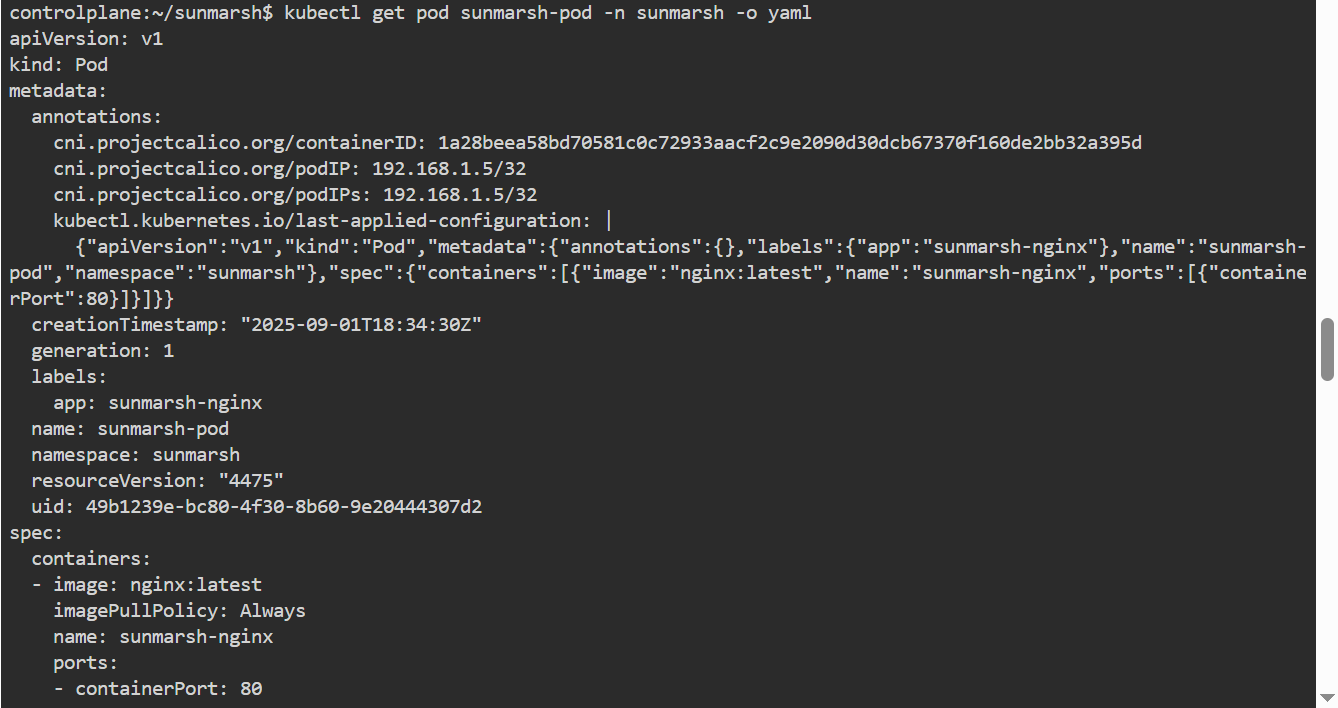
**Delete a Pod**

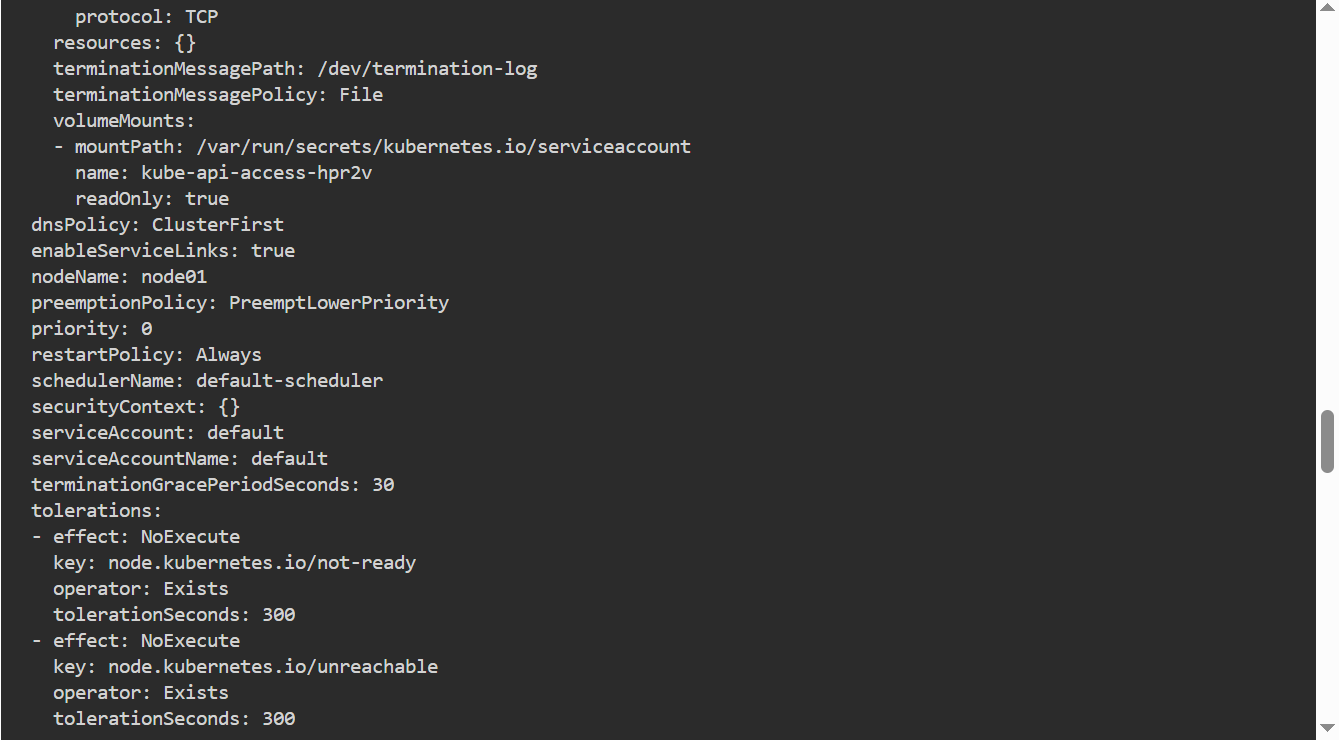
**kubectl delete pod sunmarsh-pod -n Sunmarsh**

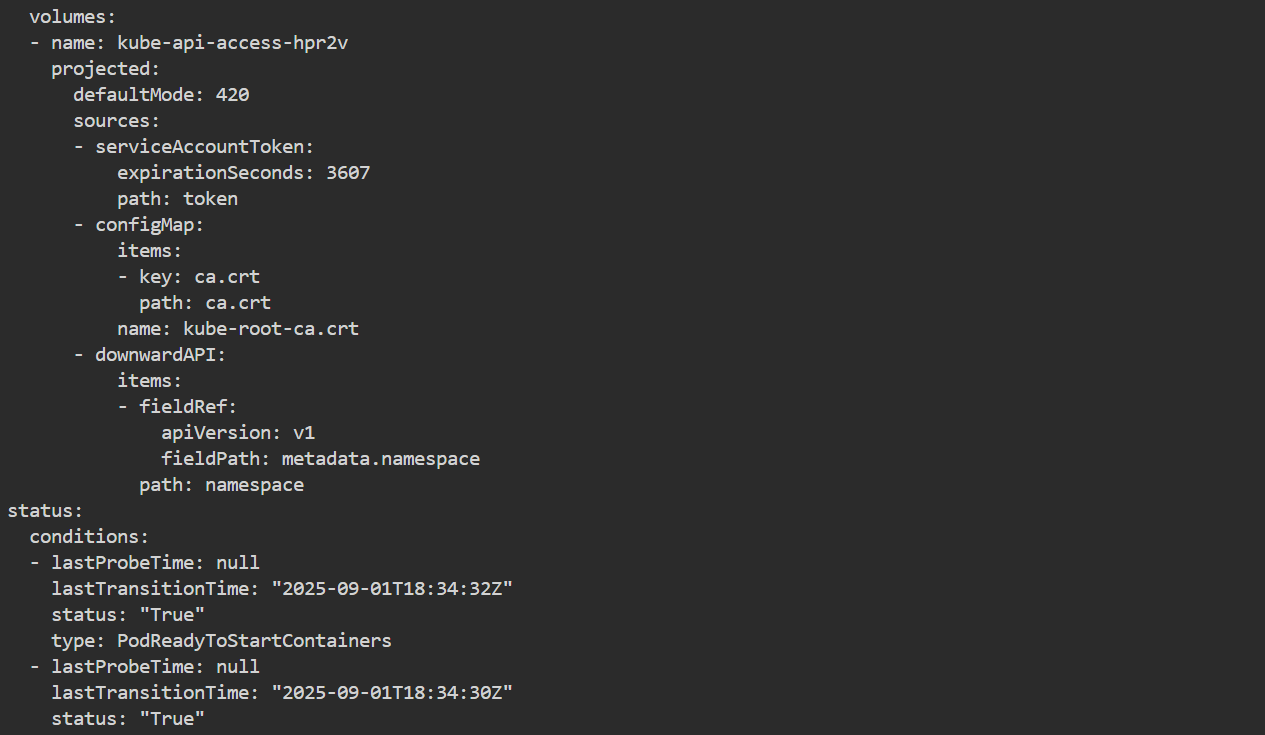
****

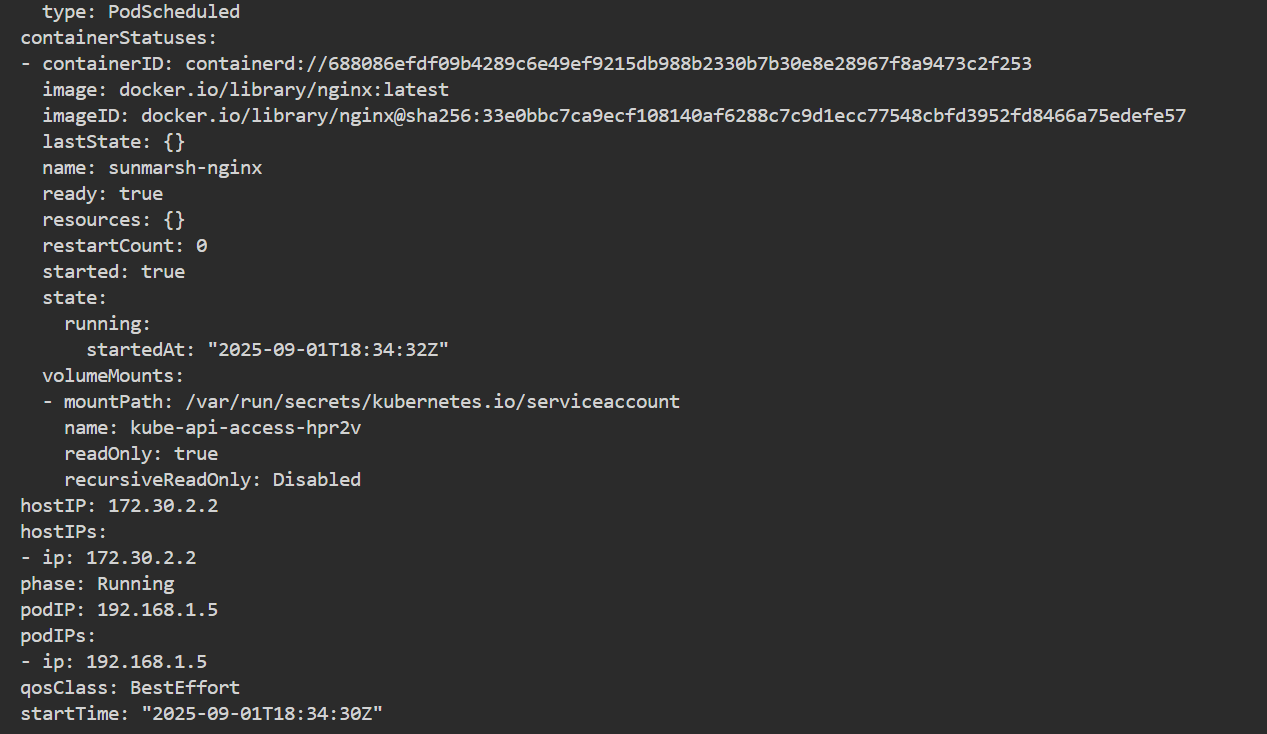
**Get Pod Status in Detail**

**kubectl get pod sunmarsh-pod -n sunmarsh -o yaml**

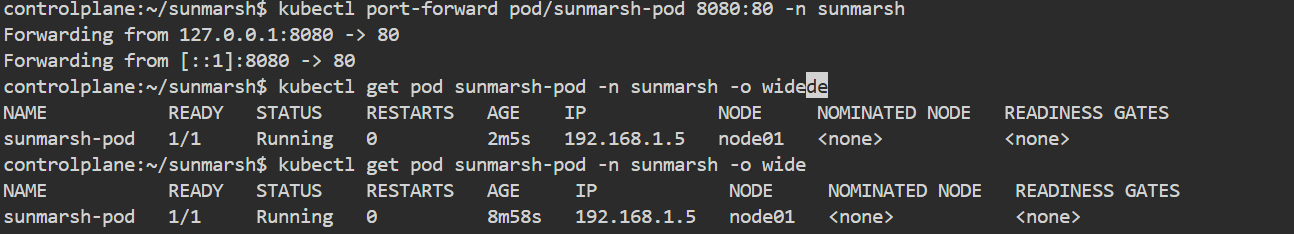
****

****

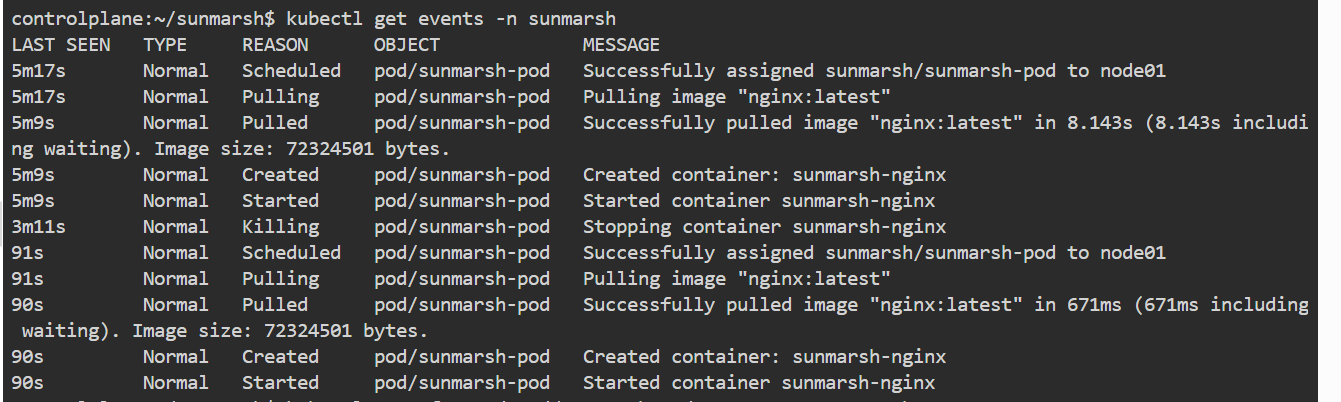
****

****

**kubectl port-forward pod/sunmarsh-pod 8080:80 -n sunmarsh**

****

**kubectl get events -n sunmarsh**

****

**ReplicaSet:**

A ReplicaSet in Kubernetes is a controller that ensures a specified number of identical Pods are always running, automatically creating or deleting Pods to maintain the desired count.

**ReplicaSet.yaml:**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

  name: sunmarsh-replica

  namespace: sunmarsh

  labels:

    app: sunmarsh-nginx-rs

spec:

  replicas: 3

  selector:

    matchLabels:

      app: sunmarsh-nginx-rs

  template:

    metadata:

      labels:

        app: sunmarsh-nginx-rs

    spec:

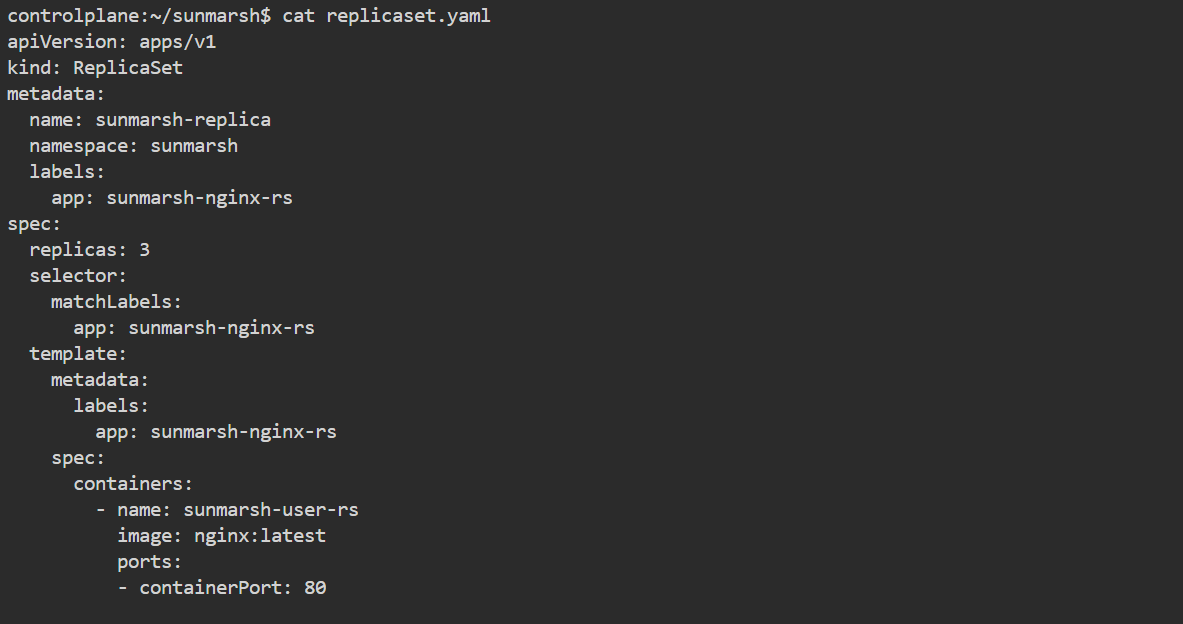
      containers:

        - name: sunmarsh-user-rs

          image: nginx:latest

          ports:

          - containerPort: 80



**1. apiVersion: apps/v1**

* ReplicaSet belongs to the **apps/v1 API group**.

**2. kind: ReplicaSet**

* Tells Kubernetes we’re creating a **ReplicaSet object**.

**3. metadata:**

* **name: sunmarsh-replica** → The name of ReplicaSet.
* **namespace: sunmarsh** → Places it in the sunmarsh namespace.
* **labels: app: sunmarsh-nginx-rs** → Adds a label (helpful for grouping/selection).

**4. spec:**

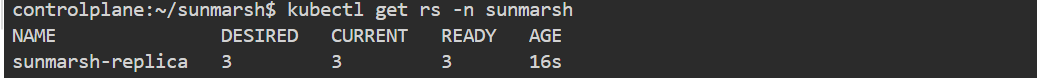
* **replicas: 3** → ReplicaSet should always run **3 Pods**.
* **selector:**
  + matchLabels: app: sunmarsh-nginx-rs  
    → ReplicaSet manages only Pods that have this label.
* **template:**
  + This is the **Pod definition** (blueprint).
  + Any Pod created will get:
    - **metadata.labels: app: sunmarsh-nginx-rs**  
      → Must match the selector, otherwise ReplicaSet won’t manage it.
    - **spec.containers:**
      * **name: sunmarsh-user-rs** → Container name.
      * **image: nginx:latest** → Runs latest NGINX web server.
      * **ports.containerPort: 80** → Exposes port 80 inside container.

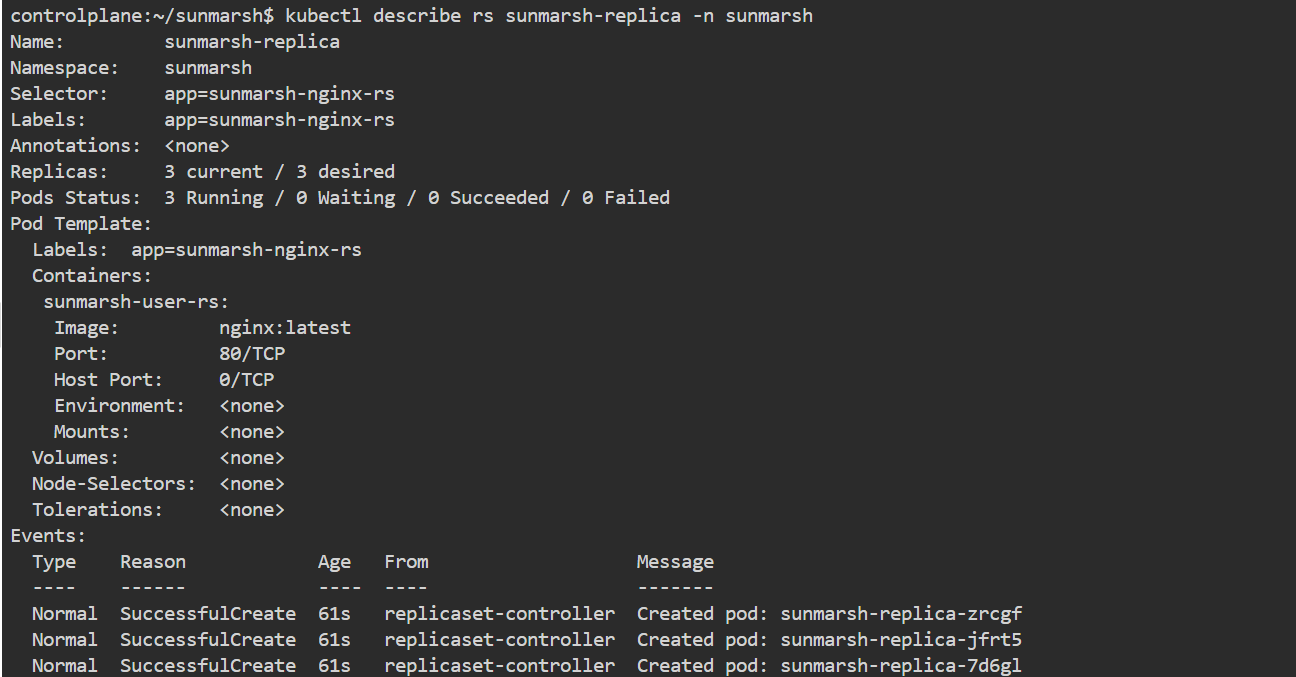
**ReplicaSet Command:**

**kubectl apply -f replicaset.yaml**

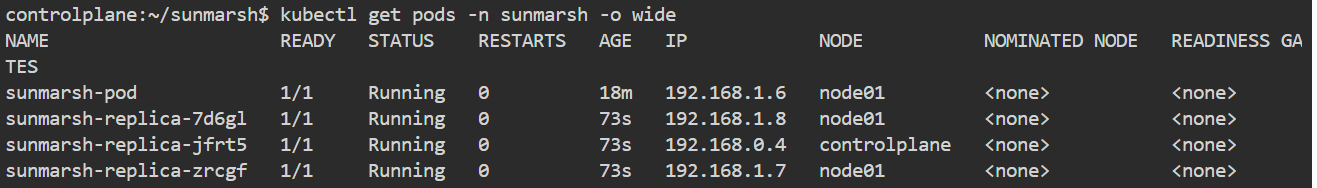
****

**kubectl get rs -n sunmarsh**

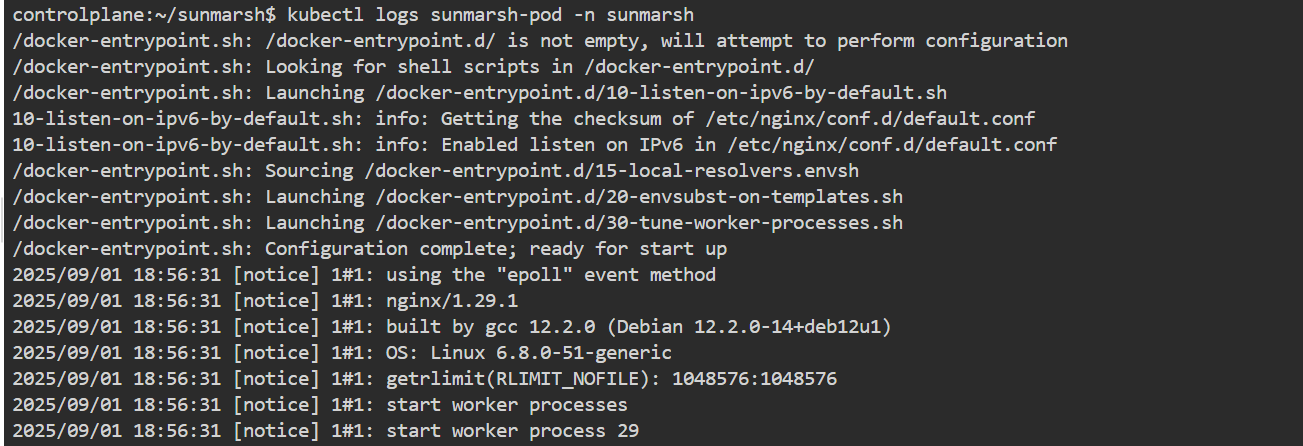
****

**kubectl describe rs sunmarsh-replica -n sunmarsh**

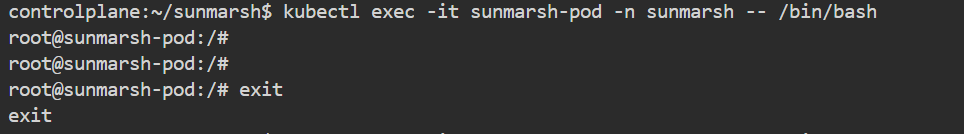
**kubectl get pods -n sunmarsh -o wide**

****

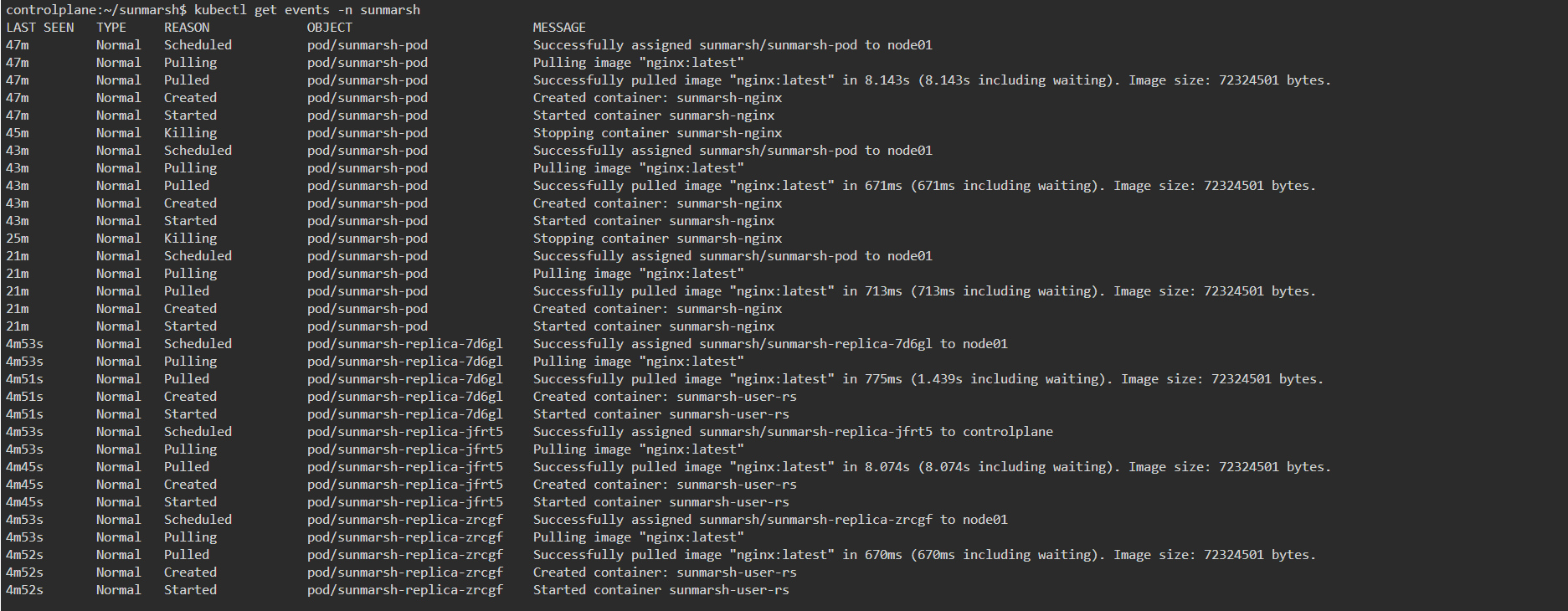
**kubectl logs <pod-name> -n Sunmarsh**

****

**kubectl exec -it <pod-name> -n sunmarsh -- /bin/bash**

****

**kubectl get events -n Sunmarsh**

****

**kubectl top pods -n Sunmarsh**

****

**kubectl delete rs sunmarsh-replica -n Sunmarsh**

****

**Deployment**

A **Deployment** in Kubernetes is a controller that manages ReplicaSets and ensures the desired number of Pods are running, while also providing rolling updates and rollbacks for application changes.

**Deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: marsh-ipl-deployment

namespace: sunmarsh

spec:

replicas: 5

selector:

matchLabels:

app: marsh-ipl

template:

metadata:

labels:

app: marsh-ipl

spec:

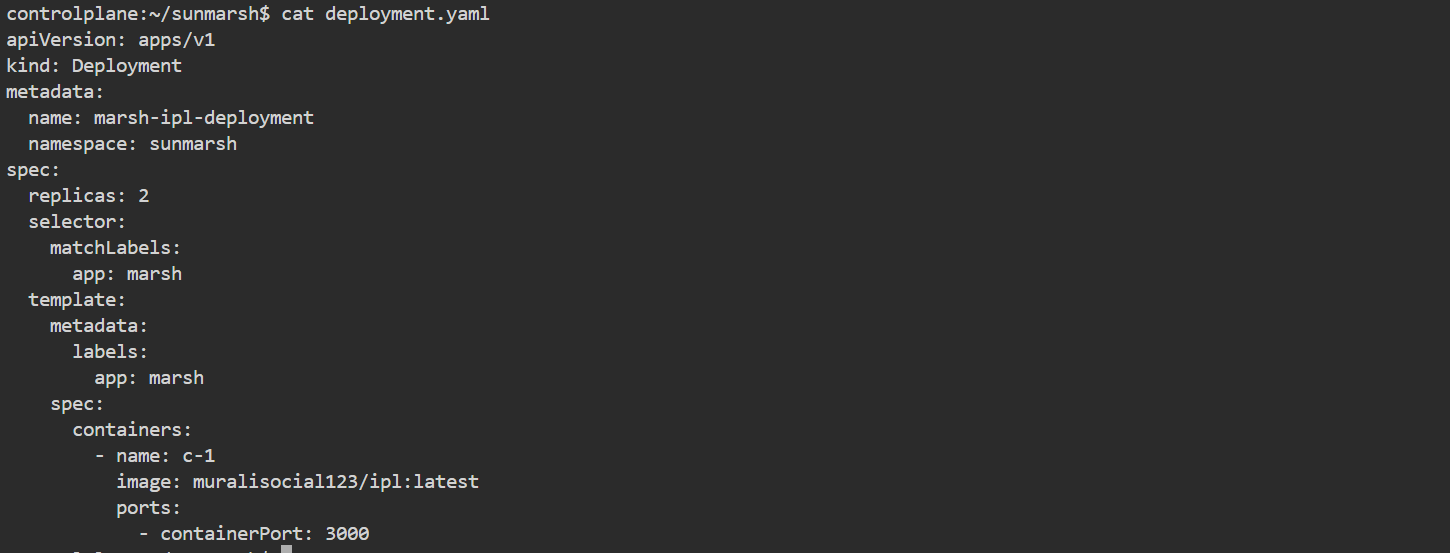
containers:

- name: c-1

image: muralisocial123/ipl:latest

ports:

- containerPort: 3000



**1. apiVersion: apps/v1**

* Deployments are part of the **apps/v1 API group**.

**2. kind: Deployment**

* This tells Kubernetes we are creating a **Deployment** object.

**3. metadata:**

* **name: marsh-ipl-deployment** → The name of the Deployment.
* **namespace: sunmarsh** → Puts this Deployment into the sunmarsh namespace.

**4. spec:**

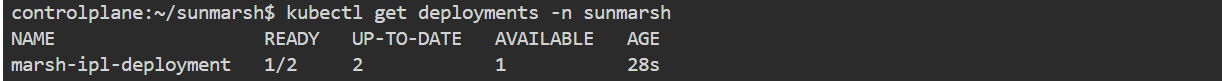
* **replicas: 2** → Kubernetes should always run **2 Pods**.
* **selector:**
  + matchLabels: app: marsh  
    → This Deployment manages Pods with the label app=marsh.
* **template:**
  + This is the **Pod blueprint** that the Deployment will use to create Pods.
  + **metadata.labels: app: marsh**  
    → Must match the selector.
* **spec.containers:**
  + **name: c-1** → Container name inside the Pod.
  + **image: muralisocial123/ipl:latest**  
    → Custom Docker image (probably a web app).
  + **ports.containerPort: 3000**  
    → Opens port 3000 inside the container (application runs here).

**Deployment command**

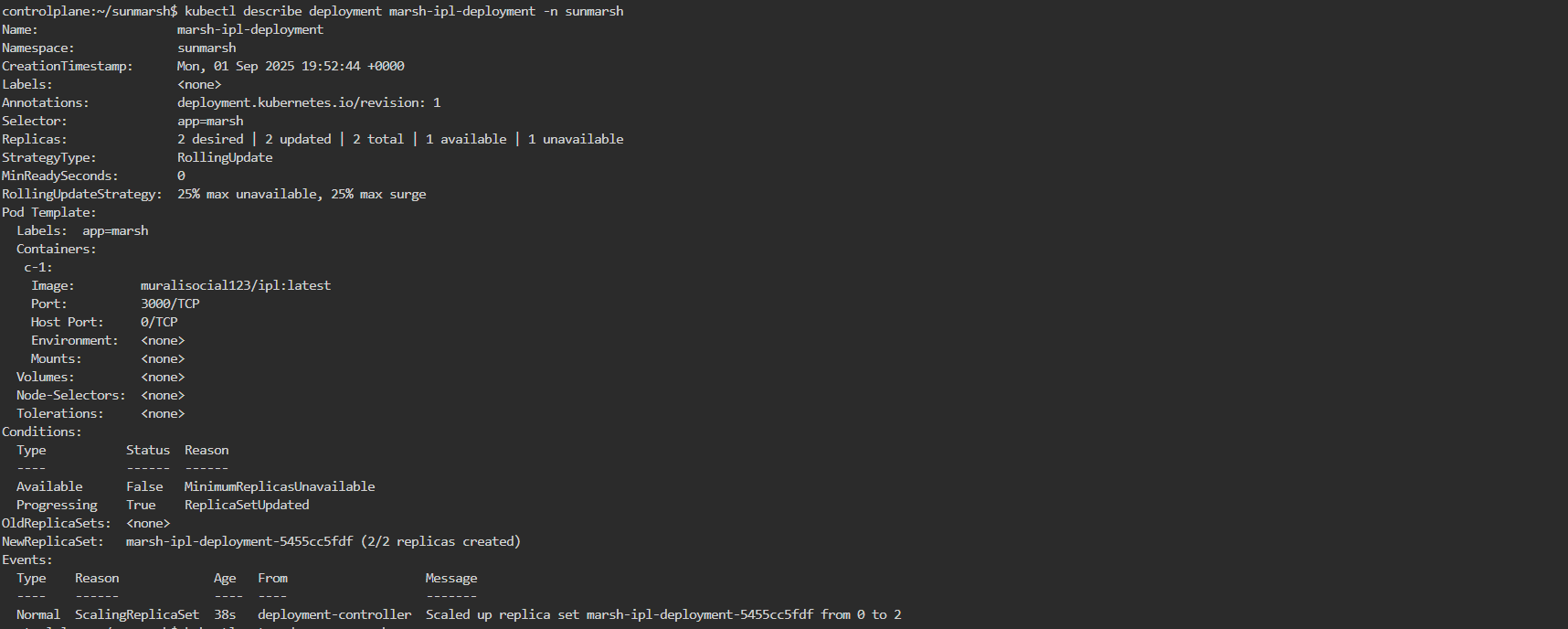
**kubectl apply -f deployment.yaml**

****

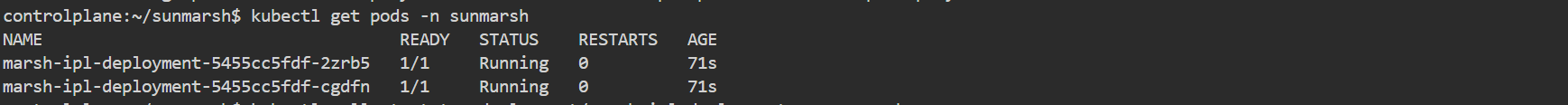
**kubectl get deployments -n sunmarsh**

****

**kubectl describe deployment marsh-ipl-deployment -n sunmarsh**

****

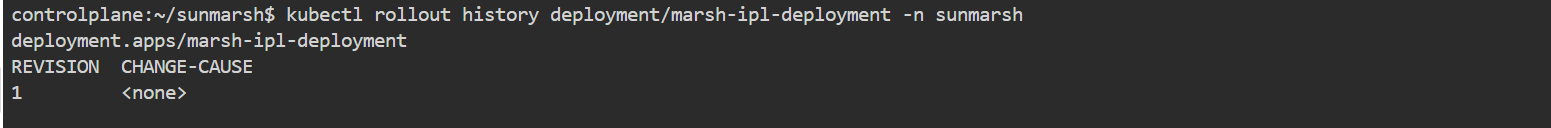
**kubectl get pods -n sunmarsh**

****

**kubectl rollout status deployment/marsh-ipl-deployment -n sunmarsh**

****

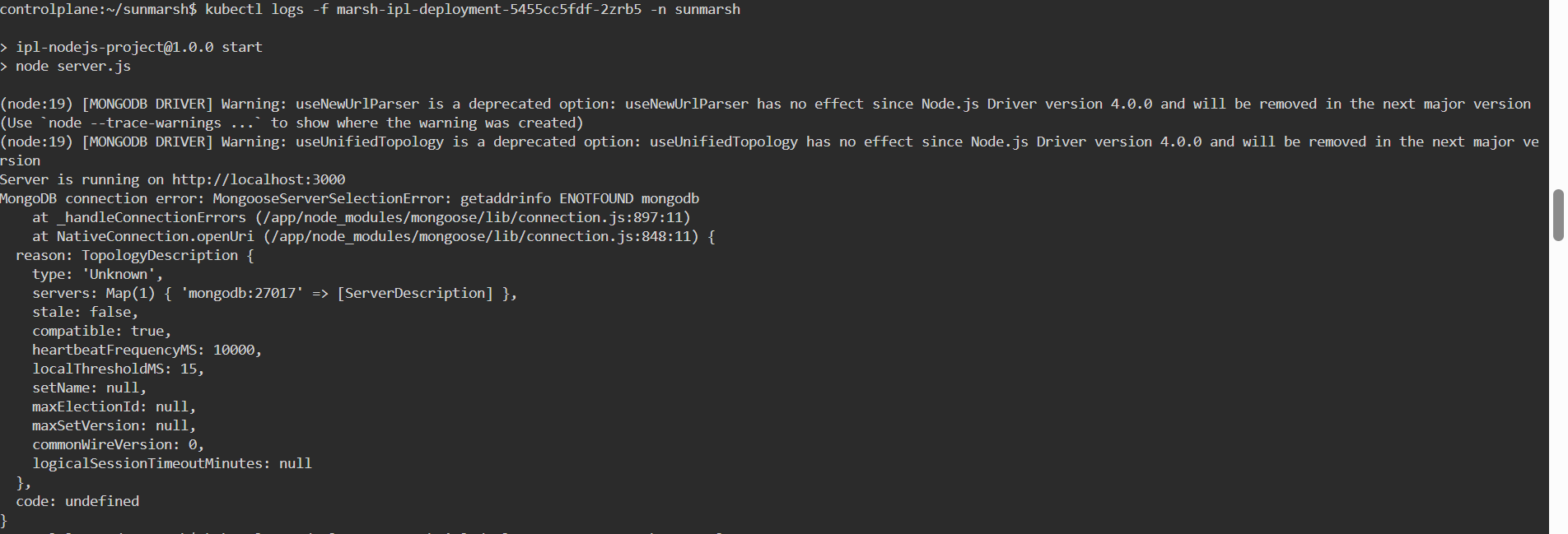
**kubectl rollout history deployment/marsh-ipl-deployment -n sunmarsh**

****

**kubectl get all -n sunmarsh**

****

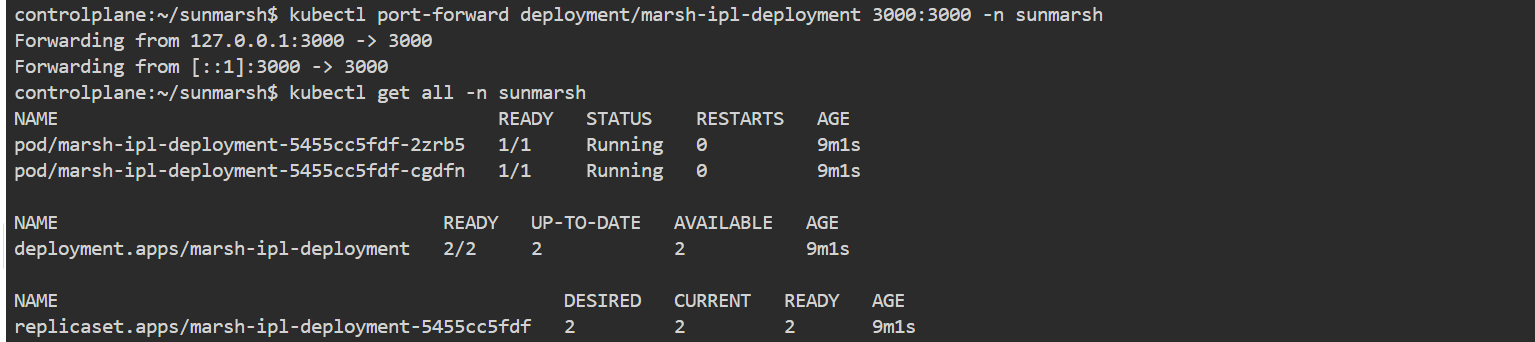
**kubectl logs -f marsh-ipl-deployment-5455cc5fdf-2zrb5 -n sunmarsh**

****

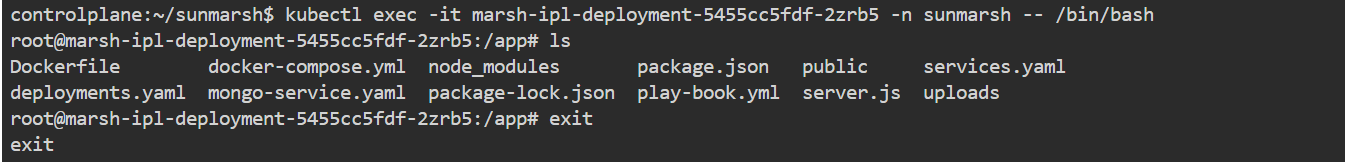
**kubectl get deployment marsh-ipl-deployment -n sunmarsh -o yaml**

****

****

**kubectl port-forward deployment/marsh-ipl-deployment 3000:3000 -n sunmarsh**

**kubectl exec -it marsh-ipl-deployment-5455cc5fdf-2zrb5 -n sunmarsh -- /bin/bash**

****

**kubectl delete deployment marsh-ipl-deployment -n sunmarsh**

****

**ClusterIP:**

ClusterIP is a Kubernetes service type that exposes an application **only within the cluster**, making it inaccessible from outside, and is typically used in production environments.

**Service- ClusterIP-Yaml**

**Service.yaml**

apiVersion: v1

kind: Service

metadata:

name: ipl-service

namespace: sunmarsh

spec:

selector:

app: marsh-ipl

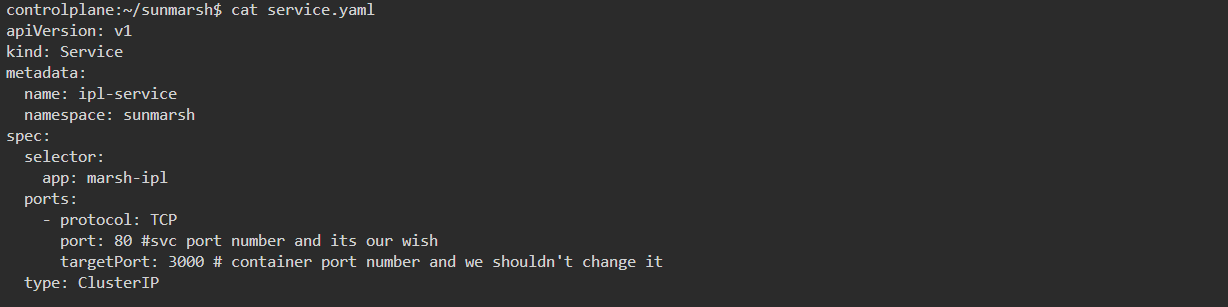
ports:

- protocol: TCP

port: 80 #svc port number and its our wish

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

type: ClusterIP



**1. apiVersion: v1**

* This tells Kubernetes which version of the API to use.
* v1 is the standard for core resources like Services, Pods, and ConfigMaps.

**2. kind: Service**

* This specifies that you are creating a **Service** resource.
* A Service allows other Pods or components to communicate with your app inside the cluster.

**3. Metadata**

metadata:

name: ipl-service

namespace: sunmarsh

* **name**: ipl-service is the name used to reference this Service inside the cluster.
* **namespace**: sunmarsh isolates this Service from resources in other namespaces.

**4. Specification (spec)**

This section defines how the Service works.

**a) Selector**

selector:

app: marsh-ipl

* Routes traffic to **Pods labeled app: marsh-ipl**.
* Only these Pods will receive requests sent to the Service.

**b) Ports**

ports:

- protocol: TCP

port: 80

targetPort: 3000

* **protocol**: TCP is the communication protocol.
* **port**: 80 is the port exposed by the Service **inside the cluster**.
* **targetPort**: 3000 is the port where the application listens inside the container.
* Requests to the Service on port 80 are forwarded to port 3000 of the Pod.

**c) Type**

type: ClusterIP

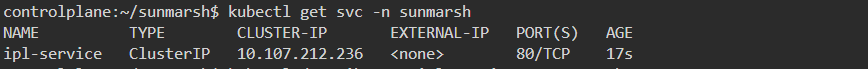
* **ClusterIP** exposes the Service **only within the cluster**.
* It is not accessible from outside the cluster.
* Commonly used for production internal communication between microservices.

**ClusterIP-Service-Command**

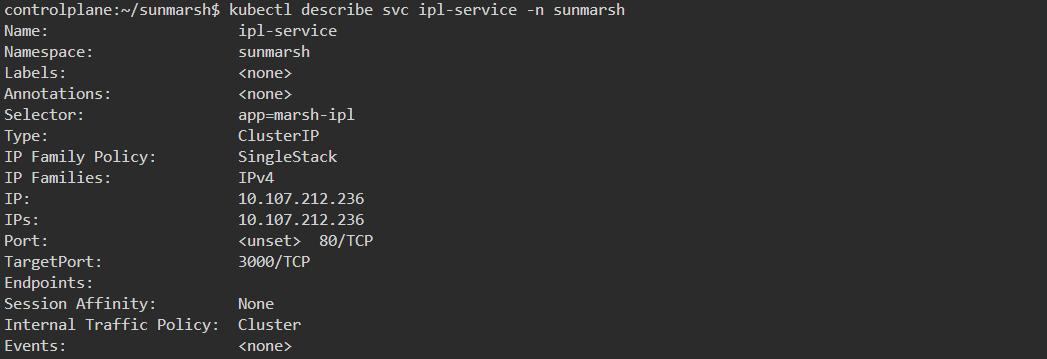
kubectl apply -f ipl-service.yaml



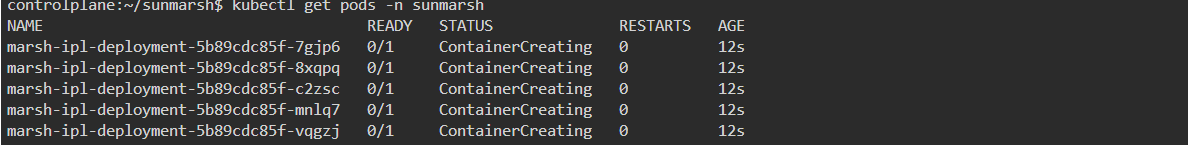
kubectl get svc -n sunmarsh



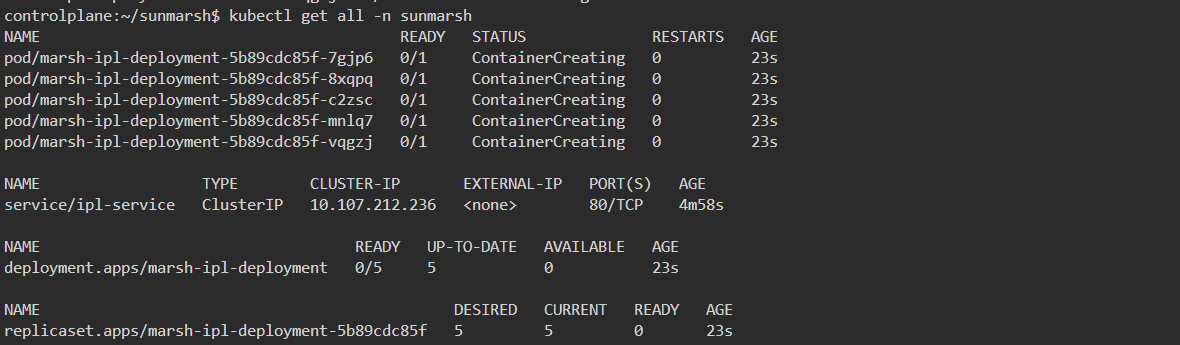
kubectl describe svc ipl-service -n sunmarsh



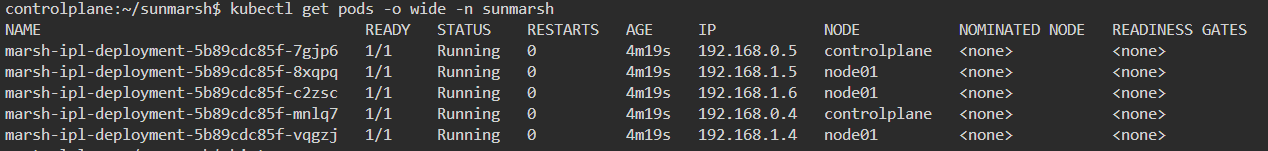
kubectl get pods -n sunmarsh



kubectl get all -n sunmarsh



kubectl get pods -o wide -n sunmarsh



**NodePort-Service**

NodePort is a Kubernetes Service type that **exposes an application on a specific port of every worker node**, allowing access both inside and outside the cluster (usually for testing), with ports limited to the range **30000–32767**.

Service.yaml

apiVersion: v1

kind: Service

metadata:

name: marsh-ipl-service

spec:

selector:

app: marsh-ipl

ports:

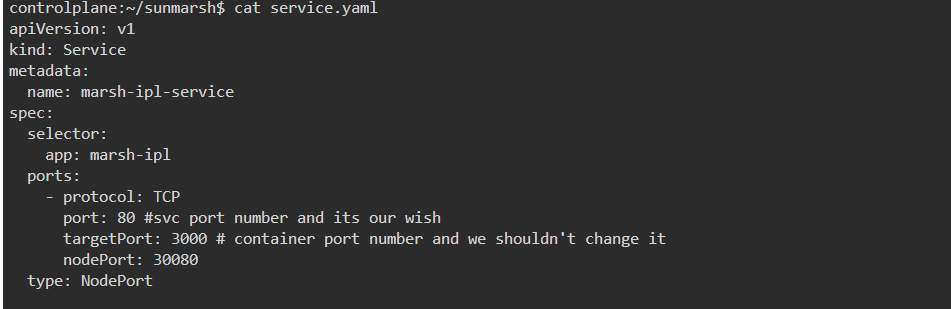
- protocol: TCP

port: 80 #svc port number and its our wish

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

nodePort: 30080

type: NodePort



**1. API Version**

apiVersion: v1

* Specifies the Kubernetes API version for this resource.
* v1 is used for core resources like Services and Pods.

**2. Kind**

kind: Service

* Declares this resource as a **Service**.
* Services allow communication to Pods inside or outside the cluster, depending on type.

**3. Metadata**

metadata:

name: marsh-ipl-service

* **name**: marsh-ipl-service is how you’ll reference this Service inside the cluster.

Note: No namespace is defined, so it will be created in the default namespace unless specified.

**4. Specification (spec)**

Defines how the Service works.

**a) Selector**

selector:

app: marsh-ipl

* Routes traffic to **Pods labeled app: marsh-ipl**.
* Only these Pods will receive requests sent to the Service.

**b) Ports**

ports:

- protocol: TCP

port: 80 # Service port inside the cluster

targetPort: 3000 # Container port to forward traffic to

nodePort: 30080 # External port on the node

* **protocol**: TCP is the communication protocol.
* **port**: Port the Service exposes **inside the cluster**.
* **targetPort**: Port where the application actually listens inside the Pod.
* **nodePort**: Port on each Node that exposes the Service externally. Traffic to this port on any Node is routed to the Service.

**c) Type**

type: NodePort

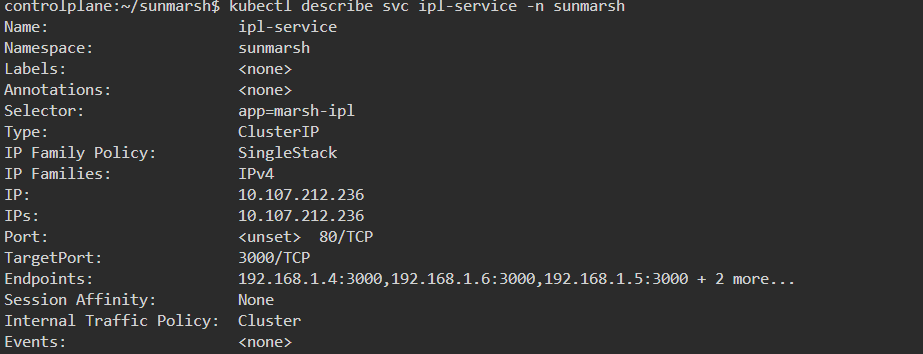
* NodePort exposes the Service **both inside and outside the cluster**.
* Makes the app accessible on <Node-IP>:<nodePort> from outside.

**NodePort Command**

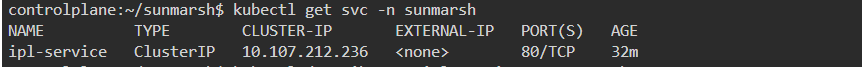
kubectl apply -f service.yaml

****

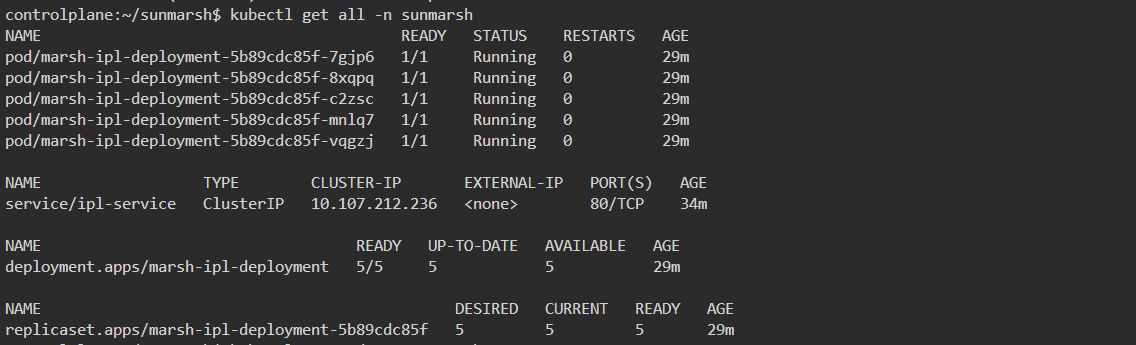
kubectl describe svc ipl-service -n sunmarsh

****

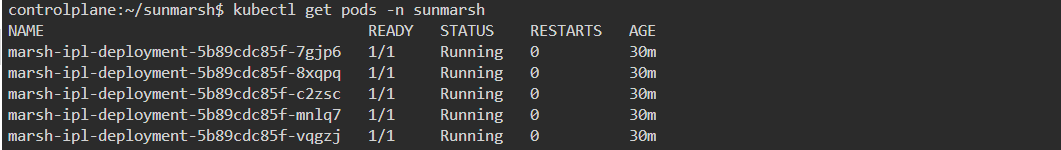
kubectl get svc -n sunmarsh

****

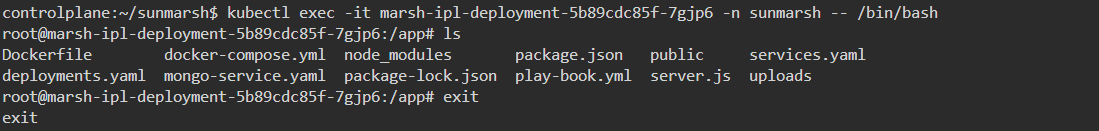
kubectl get all -n sunmarsh

****

kubectl get pods -n sunmarsh

****

kubectl exec -it marsh-ipl-deployment-5b89cdc85f-7gjp6 -n sunmarsh -- /bin/bash

****

**LoadBalancer**

A LoadBalancer (LB) in Kubernetes is a Service type that exposes an application outside the cluster using an external IP provided by a cloud provider (AWS, Azure, GCP, etc.), without port limitations, but at an additional cost, and is typically used for temporary external access without a domain name or hostname.

**LoadBalancer-service-yaml**

apiVersion: v1

kind: Service

metadata:

name: ipl-svc

namespace: sunmarsh

spec:

selector:

app: marsh-ipl

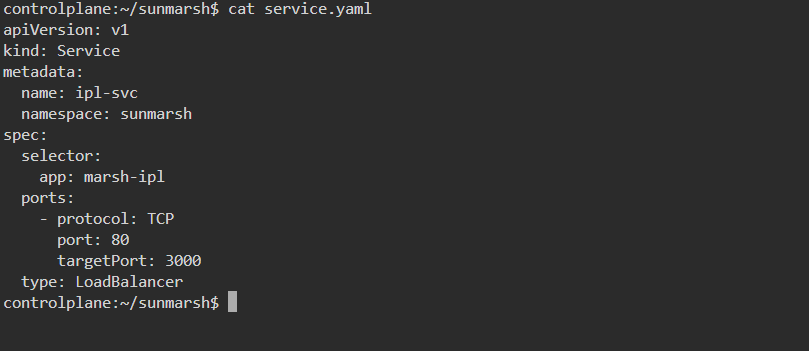
ports:

- protocol: TCP

port: 80

targetPort: 3000

type: LoadBalancer



**1. API Version**

apiVersion: v1

* Specifies the Kubernetes API version.
* v1 is used for core resources like Services and Pods.

**2. Kind**

kind: Service

* Declares that this resource is a **Service**.
* Services manage how applications are exposed inside and outside the cluster.

**3. Metadata**

metadata:

name: ipl-svc

namespace: sunmarsh

* **name**: The Service name is ipl-svc.
* **namespace**: The Service will be created inside the sunmarsh namespace.

**4. Specification (spec)**

Defines how the Service behaves.

**a) Selector**

selector:

app: marsh-ipl

* Routes traffic to Pods labeled app: marsh-ipl.
* Ensures only these Pods receive the incoming requests.

**b) Ports**

ports:

- protocol: TCP

port: 80

targetPort: 3000

* **protocol**: TCP (default communication protocol).
* **port (80)**: The port exposed by the Service.
* **targetPort (3000)**: The port inside the container where the application is running.
* Incoming traffic to port 80 will be forwarded to port 3000 on the Pod.

**c) Type**

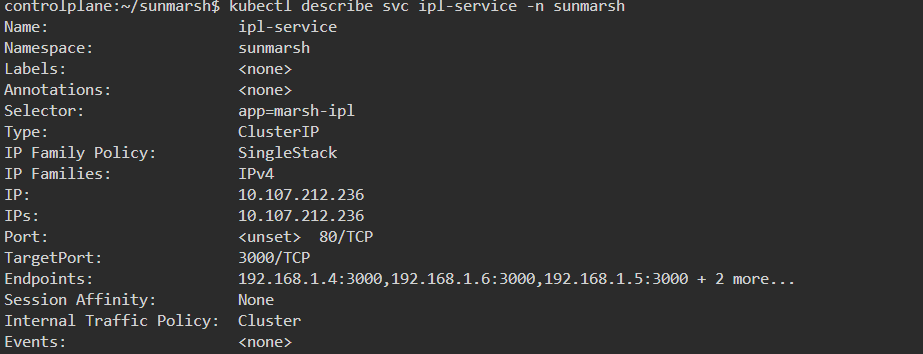
type: LoadBalancer

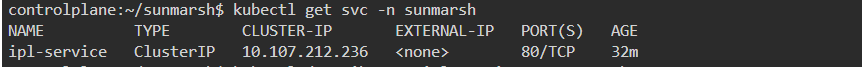
* Exposes the application **outside the cluster** through a Load Balancer.
* A **public IP address** is provided by the cloud provider (AWS, Azure, GCP, etc.).
* No port restrictions—you can use any port you want.
* Comes with **extra cost**, as cloud providers charge for Load Balancers.
* Useful when you need **direct temporary access** to your app without setting up a domain name or hostname.

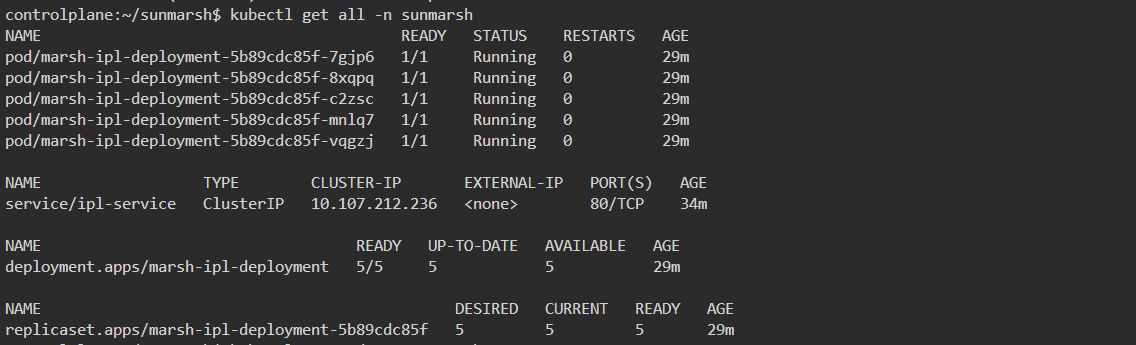
**LoadBalancer Commands**

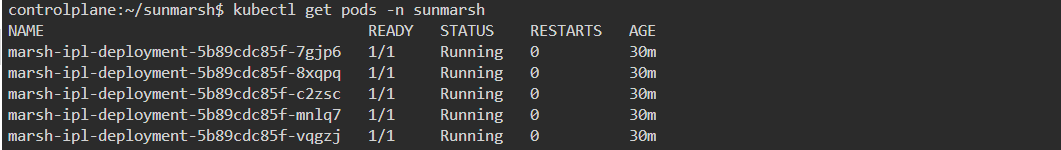
kubectl apply -f service.yaml

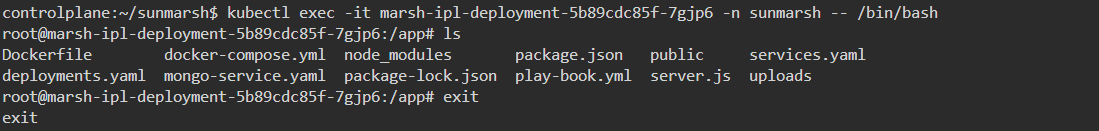
****

****

****

****

****

****

**ExternalName**

An **ExternalName** Service in Kubernetes is a specialized Service type that doesn’t route traffic to Pods but instead maps the Service to an **external DNS/hostname** (e.g., amazon.com, google.com), allowing apps to be accessed directly via their domain names.

**Service.yaml**

apiVersion: v1

kind: Service

metadata:

name: ipl-svc

namespace: sunmarsh

spec:

selector:

app: ipl

ports:

- protocol: TCP

port: 80

externalName: flipkart.com, amazon.com, google.com

type: ExternalName

