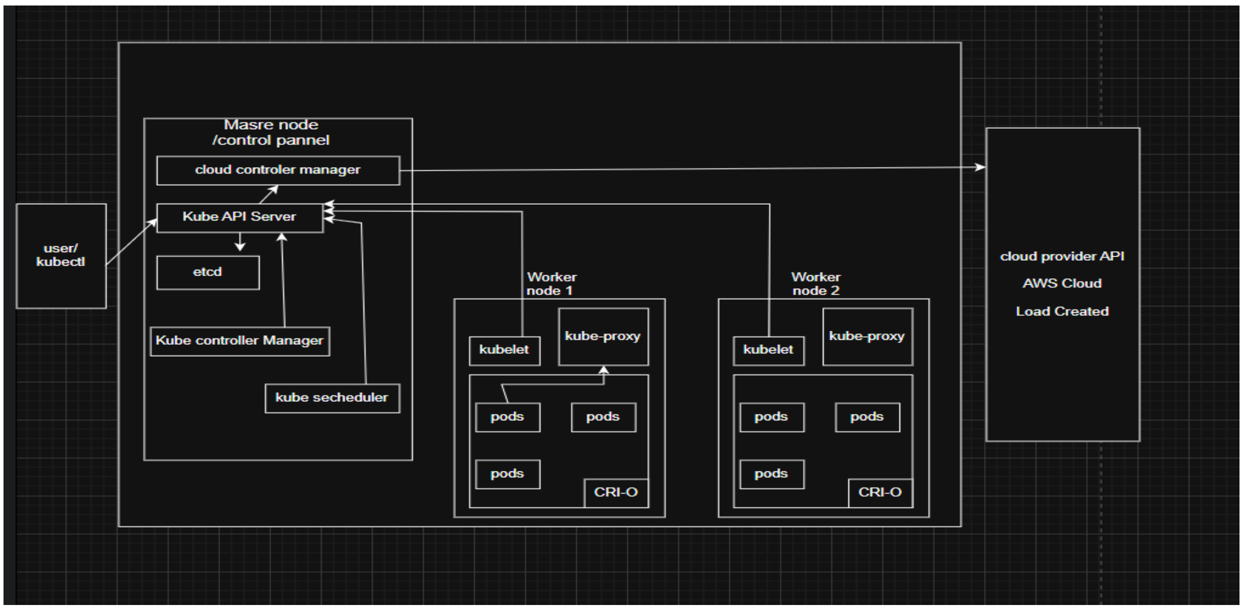
K8s document



<u>kube-apiserver :-</u>

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

<u>Etcd :-</u>

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

<u>kube-scheduler:-</u>

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

<u>kube-controller-manager :-</u>

Control plane component that runs controller processes.

<u>cloud-controller-manager :-</u>

A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.

Kubelet :-

An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

<u>kube-proxy :-</u>

kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

<u>Container runtime :-</u>

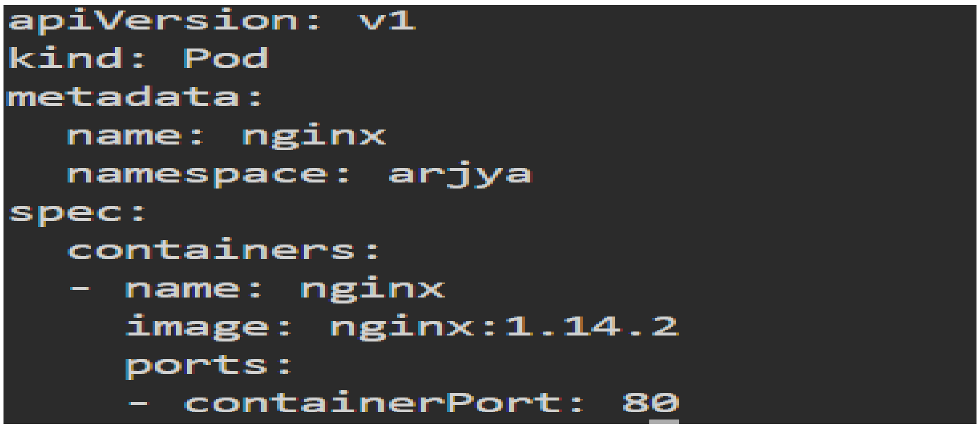
A fundamental component that empowers Kubernetes to run containers effectively. It is responsible for managing the execution and lifecycle of containers within the Kubernetes environment.

1.PODS :-

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.A *Pod* is a group of one or more containers with shared storage and network resources, and a specification for how to run the containers.

Yaml:-

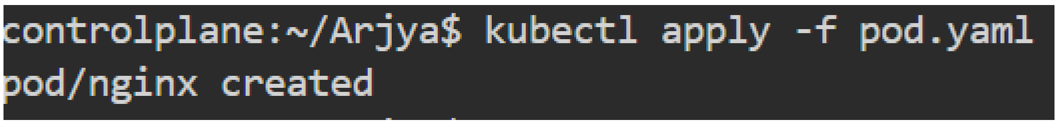
apiVersion: v1	#This tells Kubernetes which API version to use when interpreting this object
kind: Pod	#Defines the type of Kubernetes object you want to create.
metadata:	#metadata stores identifying information about the object.
name: nginx	#normal naming
namespace : arjya	#name space name
spec:	#It describes how the Pod should run.
containers:	#This field is a list, so you can define multiple containers inside the Pod
- name: nginx	#Defines the name of the container inside the Pod.
image: nginx:1.14.2	#Specifies the container image to run
ports:	#Lists the ports that this container expose
- containerPort: 80	#This container will listen on port 80



```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: arjya
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

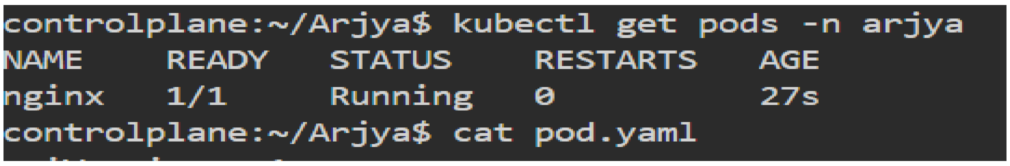
Commands:-

1.Kubectl apply -f pod.yaml : for creating pod



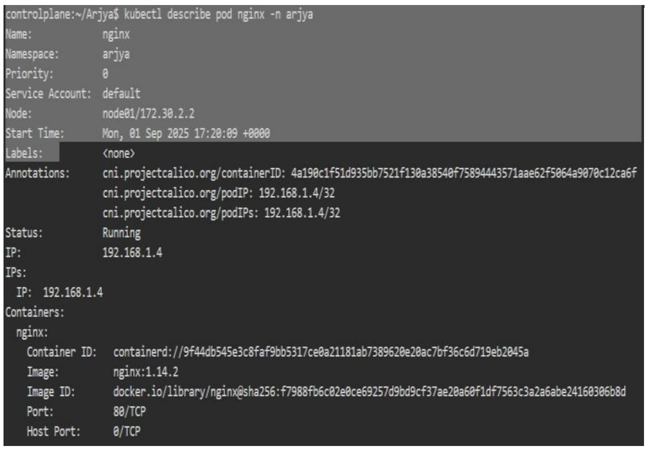
```
controlplane:~/Arjya$ kubectl apply -f pod.yaml
pod/nginx created
```

2.kubectl get pods -n arjya : for checking pods

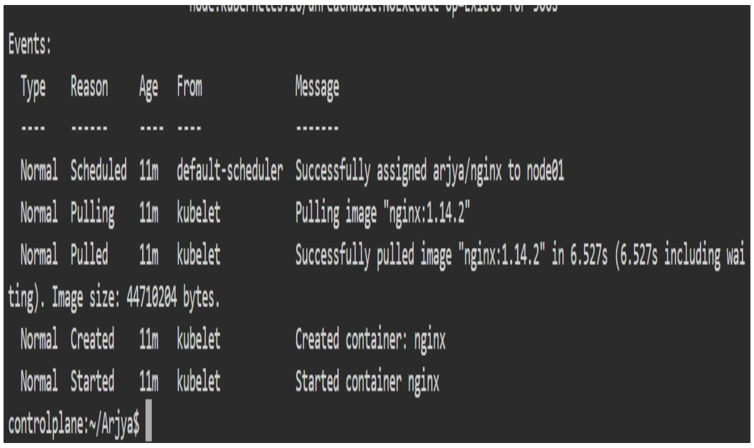


```
controlplane:~/Arjya$ kubectl get pods -n arjya
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          27s
controlplane:~/Arjya$ cat pod.yaml
```

3.kubectl get pod nginx -n arjya : to get description about pod

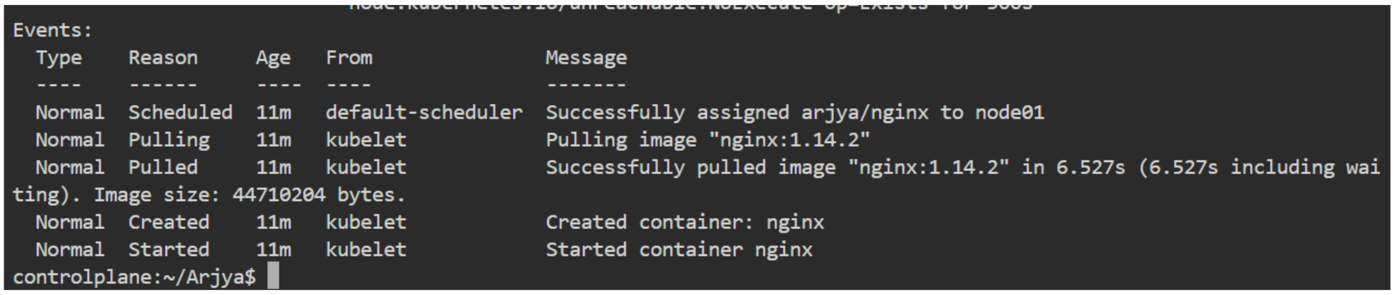


```
controlplane:~/Arjya$ kubectl describe pod nginx -n arjya
Name:             nginx
Namespace:        arjya
Priority:         0
Service Account:  default
Node:             node01/172.30.2.2
Start Time:       Mon, 01 Sep 2025 17:20:09 +0000
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: 4a190c1f51d935bb7521f130a38540f75894443571aae62f5064a9070c12ca6f
                  cni.projectcalico.org/podIP: 192.168.1.4/32
                  cni.projectcalico.org/podIPs: 192.168.1.4/32
Status:           Running
IP:               192.168.1.4
IPs:
  IP:  192.168.1.4
Containers:
  nginx:
    Container ID:   containerd://9f44db545e3c8faf9bb5317ce0a21181ab7389620e20ac7bf36c6d719eb2045a
    Image:          nginx:1.14.2
    Image ID:       docker.io/library/nginx@sha256:f7988fb6c02e0ce69257d9bd9cf37ae20a60f1df7563c3a2a6abe24160306b8d
    Port:           80/TCP
    Host Port:      0/TCP
```



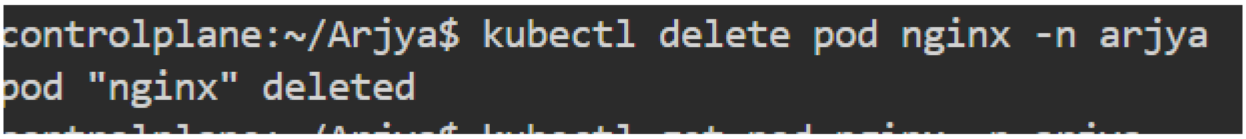
```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  11m   default-scheduler  Successfully assigned arjya/nginx to node01
  Normal  Pulling    11m   kubelet            Pulling image "nginx:1.14.2"
  Normal  Pulled     11m   kubelet            Successfully pulled image "nginx:1.14.2" in 6.527s (6.527s including wai
ting). Image size: 44710204 bytes.
  Normal  Created    11m   kubelet            Created container: nginx
  Normal  Started    11m   kubelet            Started container nginx
controlplane:~/Arjya$
```

4. **kubectl exec -it nginx -n arjya -- /bin/bash** :- Execute a command inside the Pod



```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  11m   default-scheduler  Successfully assigned arjya/nginx to node01
  Normal  Pulling    11m   kubelet            Pulling image "nginx:1.14.2"
  Normal  Pulled     11m   kubelet            Successfully pulled image "nginx:1.14.2" in 6.527s (6.527s including wai
ting). Image size: 44710204 bytes.
  Normal  Created    11m   kubelet            Created container: nginx
  Normal  Started    11m   kubelet            Started container nginx
controlplane:~/Arjya$
```

5. **kubectl delete pod nginx -n arjya** : delete pod



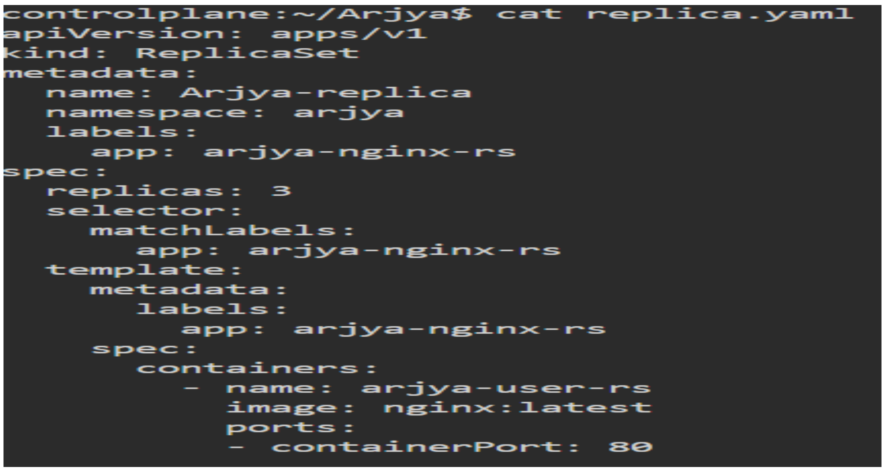
```
controlplane:~/Arjya$ kubectl delete pod nginx -n arjya
pod "nginx" deleted
```

2.Replicaset :-

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods

Yaml :-

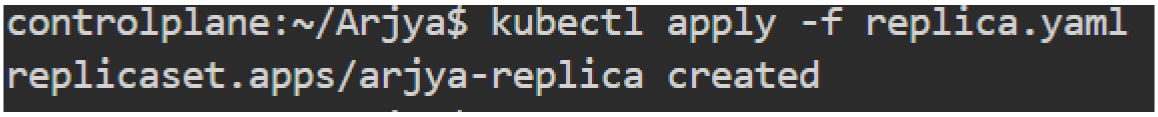
```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: Arjya-replica
  namespace: arjya
  labels:                          # key-value pairs attached to this ReplicaSet. They help identify/select resources.
    app: arjya-nginx-rs
spec:                              #  spec defines the desired state.
  replicas: 3                      # means the ReplicaSet will maintain 3 Pods
  selector:                        # Defines how the ReplicaSet knows which Pods belong to it.
    matchLabels:
      app: arjya-nginx-rs          # This must match the labels in the Pod template below.
  template:                        # This pod templete Any new Pod created by this ReplicaSet will use this template.
    metadata:
      labels:
        app: arjya-nginx-rs
    spec:
      containers:
        - name: arjya-user-rs
          image: nginx:latest
          ports:
          - containerPort: 80      # the Nginx container listens on port 80
```



```
controlplane:~/Arjya$ cat replica.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: Arjya-replica
  namespace: arjya
  labels:
    app: arjya-nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: arjya-nginx-rs
  template:
    metadata:
      labels:
        app: arjya-nginx-rs
    spec:
      containers:
        - name: arjya-user-rs
          image: nginx:latest
          ports:
          - containerPort: 80
```

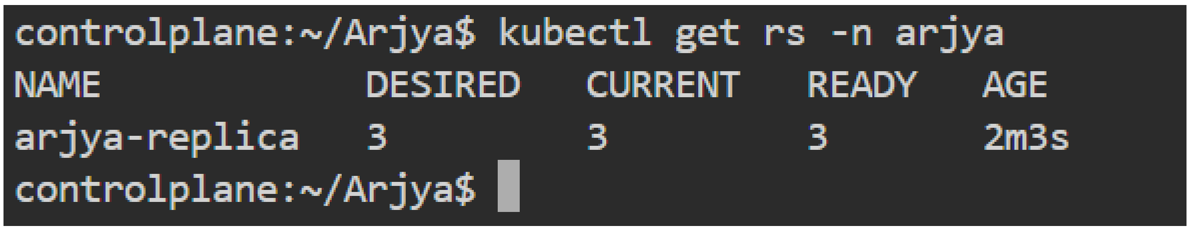
Commands:-

1. **kubectl apply -f replica.yaml :-**create replica



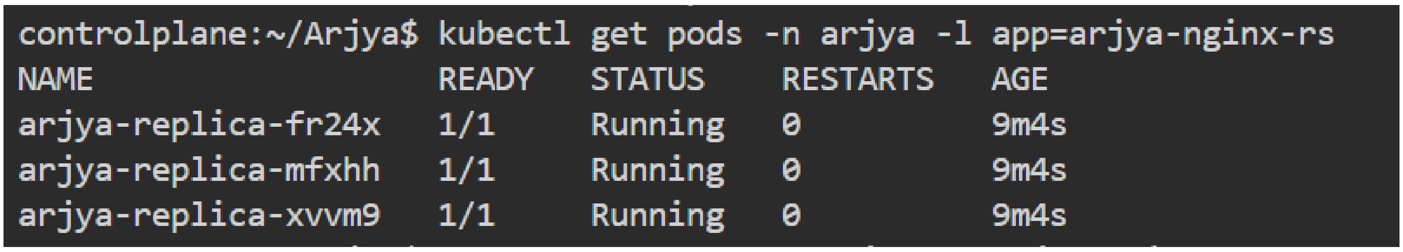
```
controlplane:~/Arjya$ kubectl apply -f replica.yaml
replicaset.apps/arjya-replica created
```

2. **kubectl get rs -n arjya:** check replicaset created or not



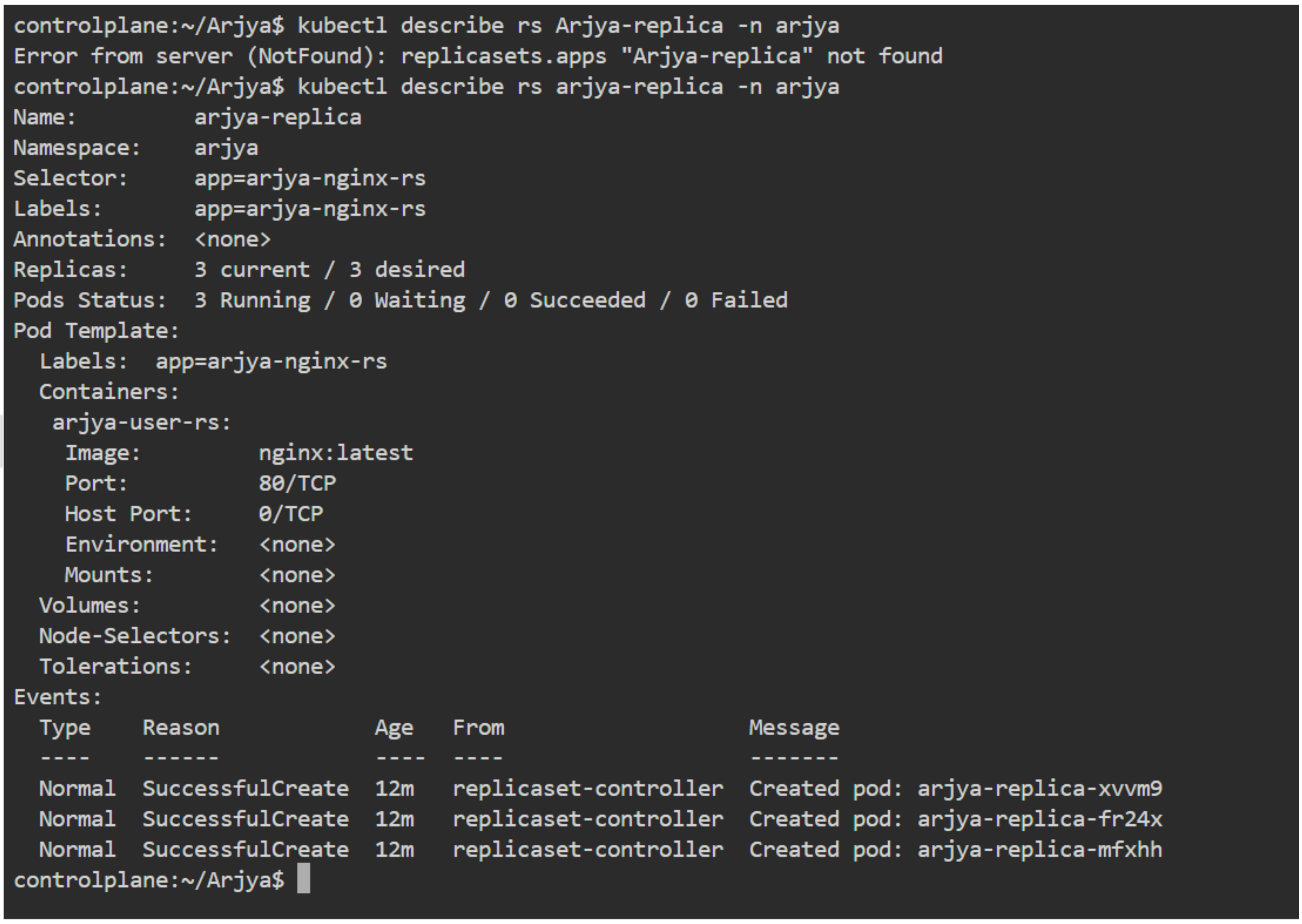
```
controlplane:~/Arjya$ kubectl get rs -n arjya
NAME            DESIRED   CURRENT   READY   AGE
arjya-replica   3         3         3       2m3s
controlplane:~/Arjya$
```

3. **ds -n arjya -l app=arjya-nginx-rs :** list all replicas



```
controlplane:~/Arjya$ kubectl get pods -n arjya -l app=arjya-nginx-rs
NAME                  READY   STATUS    RESTARTS   AGE
arjya-replica-fr24x   1/1     Running   0          9m4s
arjya-replica-mfxhh   1/1     Running   0          9m4s
arjya-replica-xvvm9   1/1     Running   0          9m4s
```

4. **kubectl describe rs Arjya-replica -n arjya :** describe replicaset



```
controlplane:~/Arjya$ kubectl describe rs Arjya-replica -n arjya
Error from server (NotFound): replicasets.apps "Arjya-replica" not found
controlplane:~/Arjya$ kubectl describe rs arjya-replica -n arjya
Name:         arjya-replica
Namespace:    arjya
Selector:     app=arjya-nginx-rs
Labels:       app=arjya-nginx-rs
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=arjya-nginx-rs
  Containers:
   arjya-user-rs:
    Image:        nginx:latest
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
  Node-Selectors: <none>
  Tolerations:    <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  12m   replicaset-controller  Created pod: arjya-replica-xvvm9
  Normal  SuccessfulCreate  12m   replicaset-controller  Created pod: arjya-replica-fr24x
  Normal  SuccessfulCreate  12m   replicaset-controller  Created pod: arjya-replica-mfxhh
controlplane:~/Arjya$
```

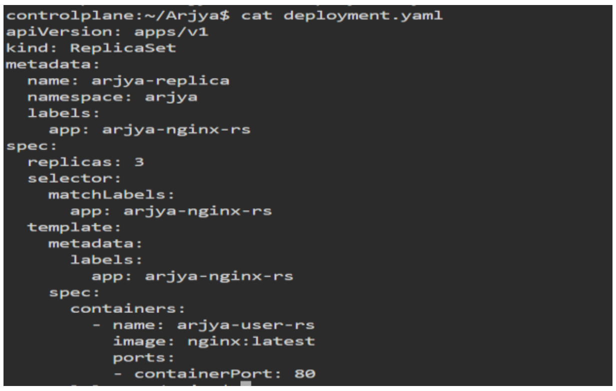
Deployment:-

A *Deployment* provides declarative updates for pods and replicaset. Create a Deployment to rollout a ReplicaSet. The ReplicaSet creates Pods in the background. Check the status of the rollout to see if it succeeds or not.

Yaml :-

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: Arjya-dp
  namespace: arjya
  labels:
    app: arjya-nginx-dp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: arjya-nginx-dp
  template:
    metadata:
      labels:
        app: arjya-nginx-dp
    spec:
      containers:
        - name: arjya-user-dp
          image: nginx:1.21.3
          ports:
          - containerPort: 80
```

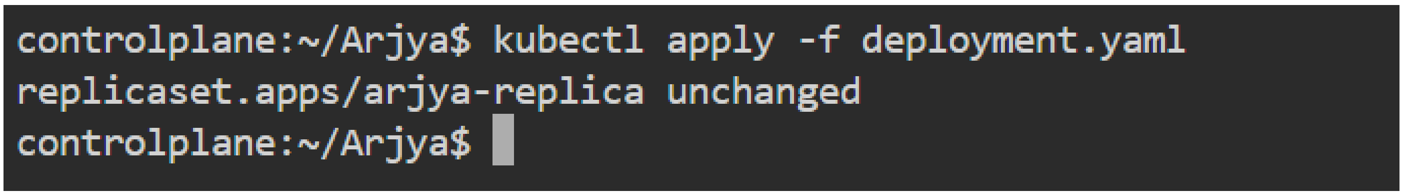
1.



```
controlplane:~/Arjya$ cat deployment.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: arjya-replica
  namespace: arjya
  labels:
    app: arjya-nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: arjya-nginx-rs
  template:
    metadata:
      labels:
        app: arjya-nginx-rs
    spec:
      containers:
        - name: arjya-user-rs
          image: nginx:latest
          ports:
          - containerPort: 80
```

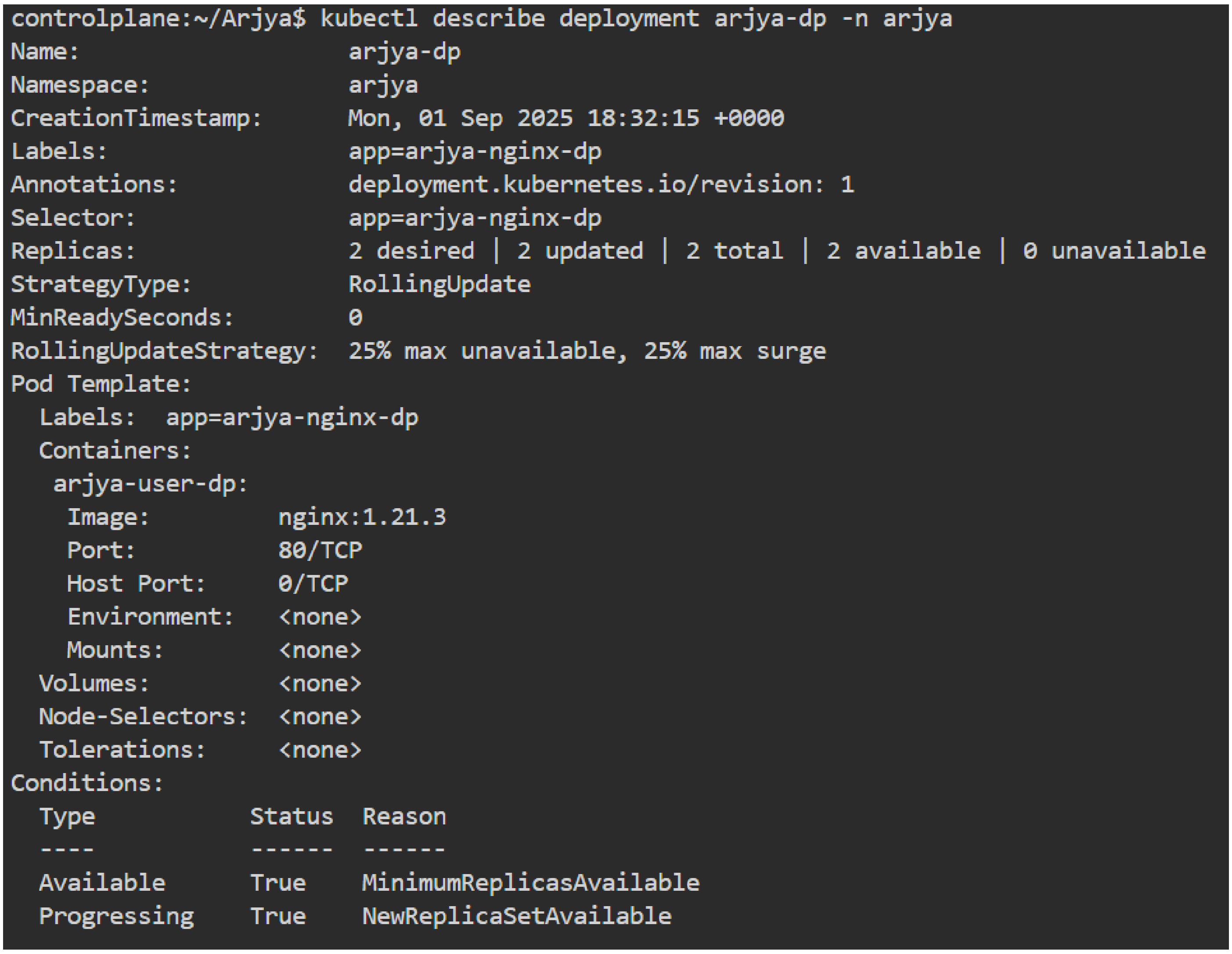
Commands:-

1.ubectl apply -f deployment.yaml :- to create deployment



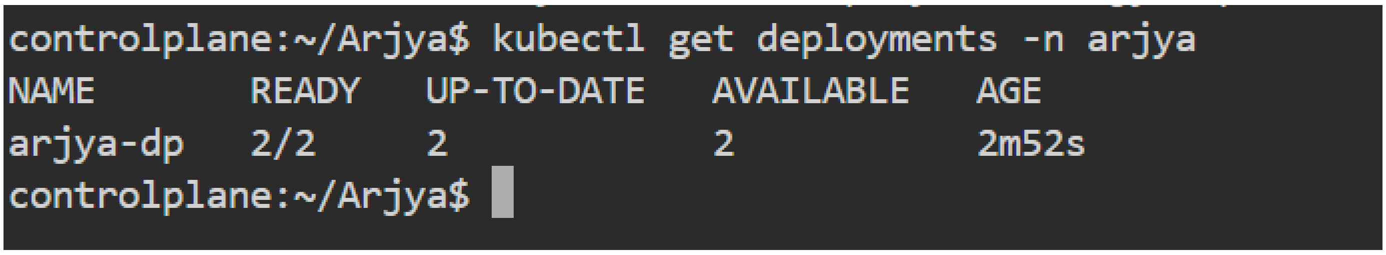
```
controlplane:~/Arjya$ kubectl apply -f deployment.yaml
replicaset.apps/arjya-replica unchanged
controlplane:~/Arjya$
```

2. kubectl describe deployment Arjya-dp -n arjya : describe deployment



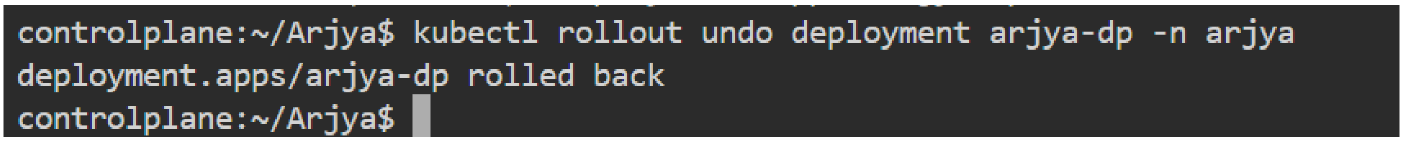
```
controlplane:~/Arjya$ kubectl describe deployment arjya-dp -n arjya
Name:                   arjya-dp
Namespace:              arjya
CreationTimestamp:      Mon, 01 Sep 2025 18:32:15 +0000
Labels:                 app=arjya-nginx-dp
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=arjya-nginx-dp
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=arjya-nginx-dp
  Containers:
   arjya-user-dp:
    Image:         nginx:1.21.3
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
```

5. **kubectl get deployments -n arjya : to see deployment**



```
controlplane:~/Arjya$ kubectl get deployments -n arjya
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
arjya-dp   2/2     2            2           2m52s
controlplane:~/Arjya$
```

6. **kubectl rollout undo deployment arjya-dp -n arjya : rollout the task (image)**



```
controlplane:~/Arjya$ kubectl rollout undo deployment arjya-dp -n arjya
deployment.apps/arjya-dp rolled back
controlplane:~/Arjya$
```

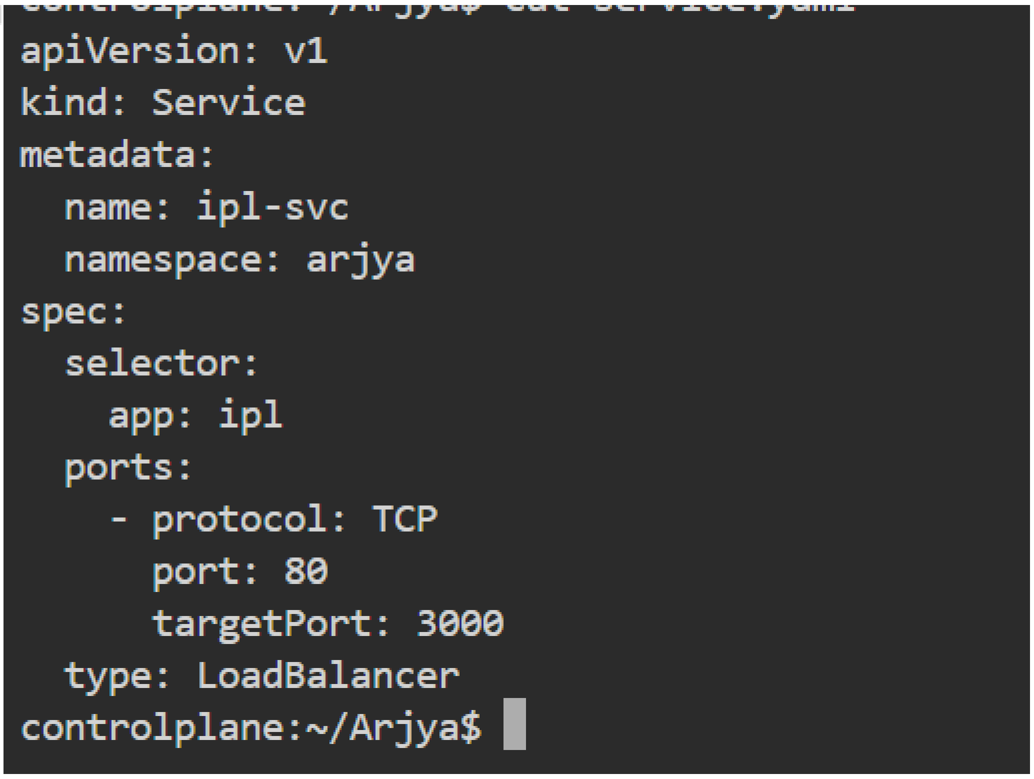
Service:-

A Service in Kubernetes is an abstraction layer that provides a stable network endpoint to access a set of Pods.

Type:- ClusterIP, NodePort, LoadBalancer, ExternalName

Yaml :-

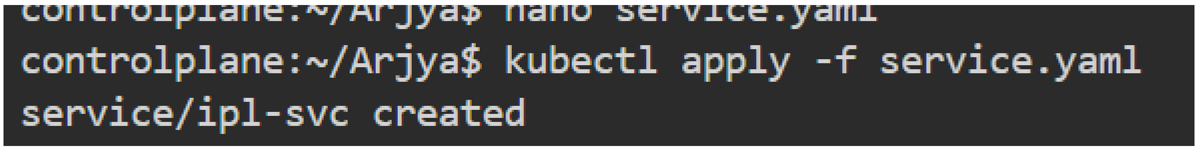
```
apiversion: v1
kind: Service
metadata:
  name: arjya-service
  namespace: arjya
spec:
  selector:
    app: arjya-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
  type: LoadBalancer
```



```
apiVersion: v1
kind: Service
metadata:
  name: ipl-svc
  namespace: arjya
spec:
  selector:
    app: ipl
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
  type: LoadBalancer
controlplane:~/Arjya$
```

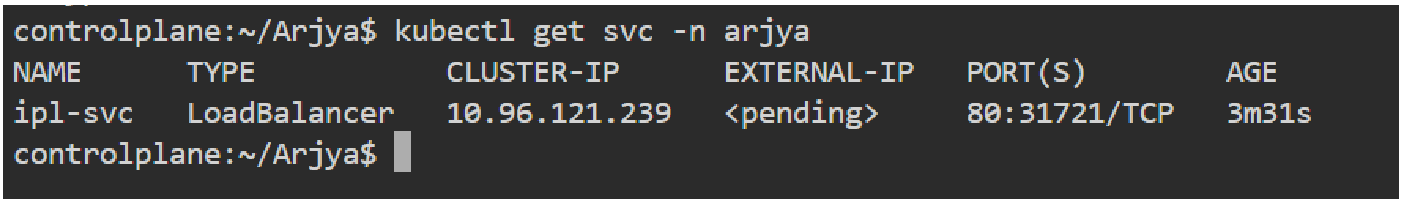
Commands:-

1. **kubectl apply -f service.yaml:** create sevice



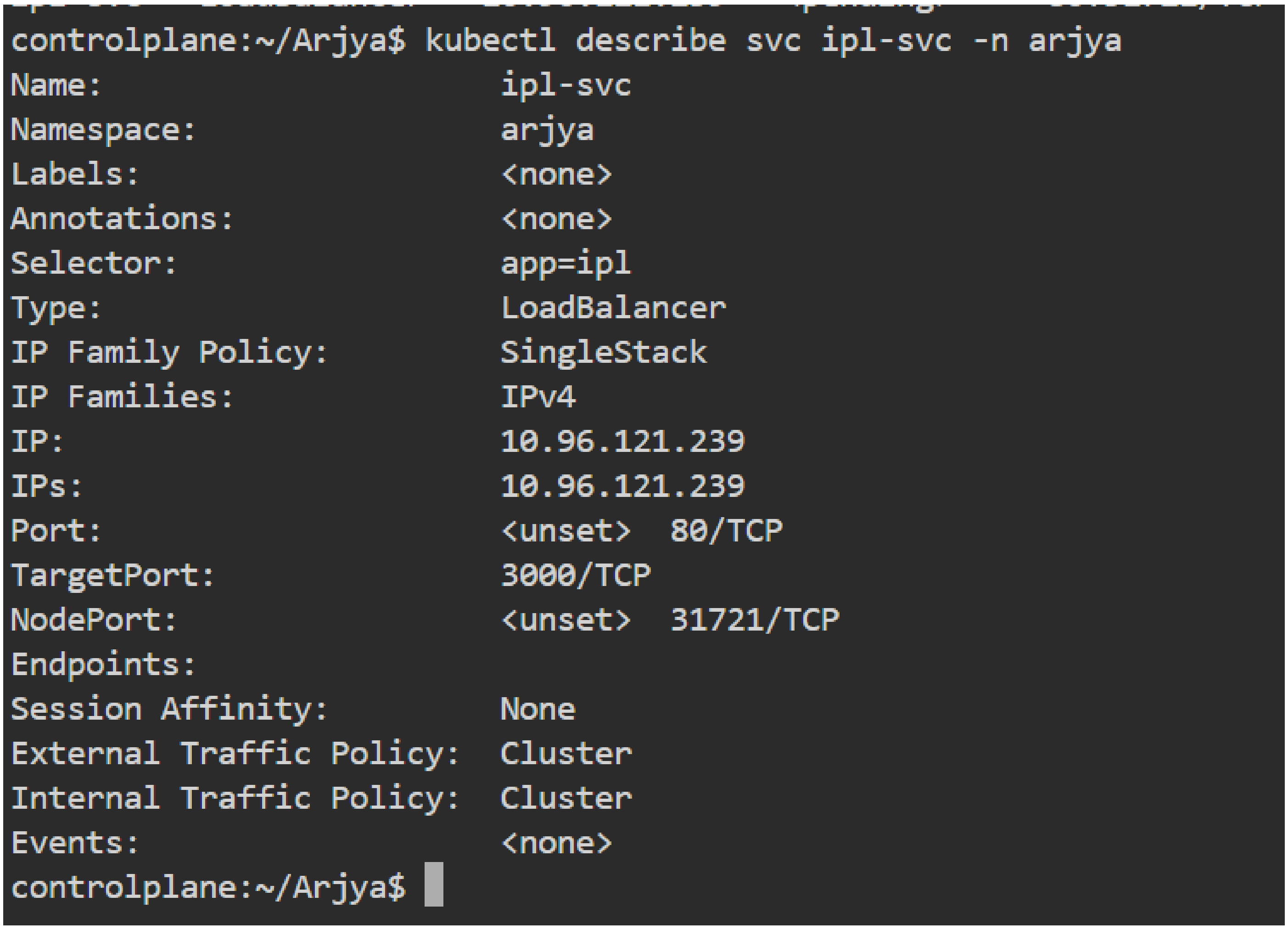
```
controlplane:~/Arjya$ nano service.yaml
controlplane:~/Arjya$ kubectl apply -f service.yaml
service/ipl-svc created
```

2. **kubectl get svc -n arjya :** check the service



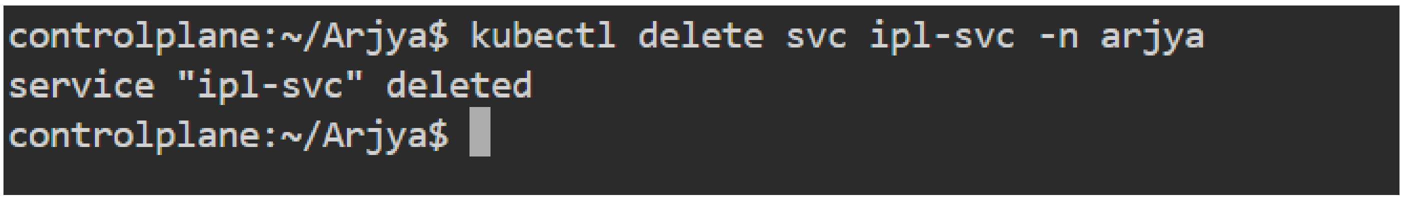
```
controlplane:~/Arjya$ kubectl get svc -n arjya
NAME      TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
ipl-svc   LoadBalancer   10.96.121.239   <pending>     80:31721/TCP   3m31s
controlplane:~/Arjya$
```

3. **kubectl describe svc ipl-svc -n arjya**: describe service



```
controlplane:~/Arjya$ kubectl describe svc ipl-svc -n arjya
Name:                     ipl-svc
Namespace:                arjya
Labels:                   <none>
Annotations:              <none>
Selector:                 app=ipl
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.96.121.239
IPs:                      10.96.121.239
Port:                     <unset>  80/TCP
TargetPort:               3000/TCP
NodePort:                 <unset>  31721/TCP
Endpoints:
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
controlplane:~/Arjya$
```

4. **kubectl delete svc ipl-svc -n arjya :- delete service**



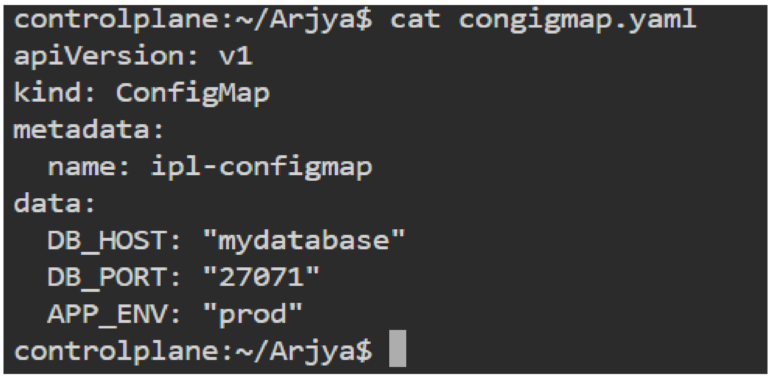
```
controlplane:~/Arjya$ kubectl delete svc ipl-svc -n arjya
service "ipl-svc" deleted
controlplane:~/Arjya$
```

ConfigMaps :-

A ConfigMap is an API object used to store non-confidential data in key-value pairs.

Yaml :-

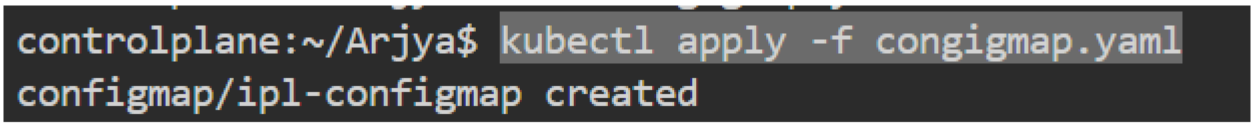
```
apiVersion: v1

kind: ConfigMap

metadata:

  name: ipl-configmap

data:

  DB_HOST: "mydatabase"                    # Database host is set to mydatabase

  DB_PORT: "27071"                         # Database port is set to 27071.

  APP_ENV: "prod"                          # Environment is set to production.
```



```
controlplane:~/Arjya$ cat congigmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ipl-configmap
data:
  DB_HOST: "mydatabase"
  DB_PORT: "27071"
  APP_ENV: "prod"
controlplane:~/Arjya$
```

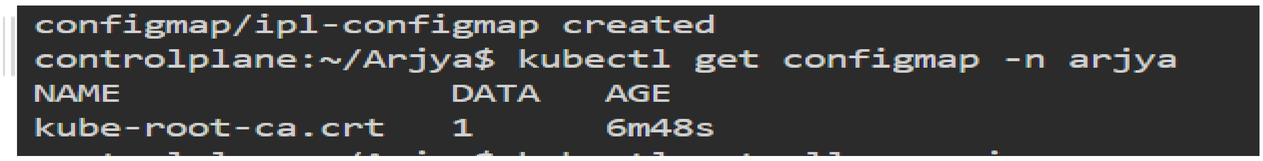
Commands :-

1.kubectl apply -f congigmap.yaml :-for creating configmap



```
controlplane:~/Arjya$ kubectl apply -f congigmap.yaml
configmap/ipl-configmap created
```

2. kubectl get configmap -n arjya :- to check configmap created or not



```
configmap/ipl-configmap created
controlplane:~/Arjya$ kubectl get configmap -n arjya
NAME                    DATA    AGE
kube-root-ca.crt        1       6m48s
```

3.