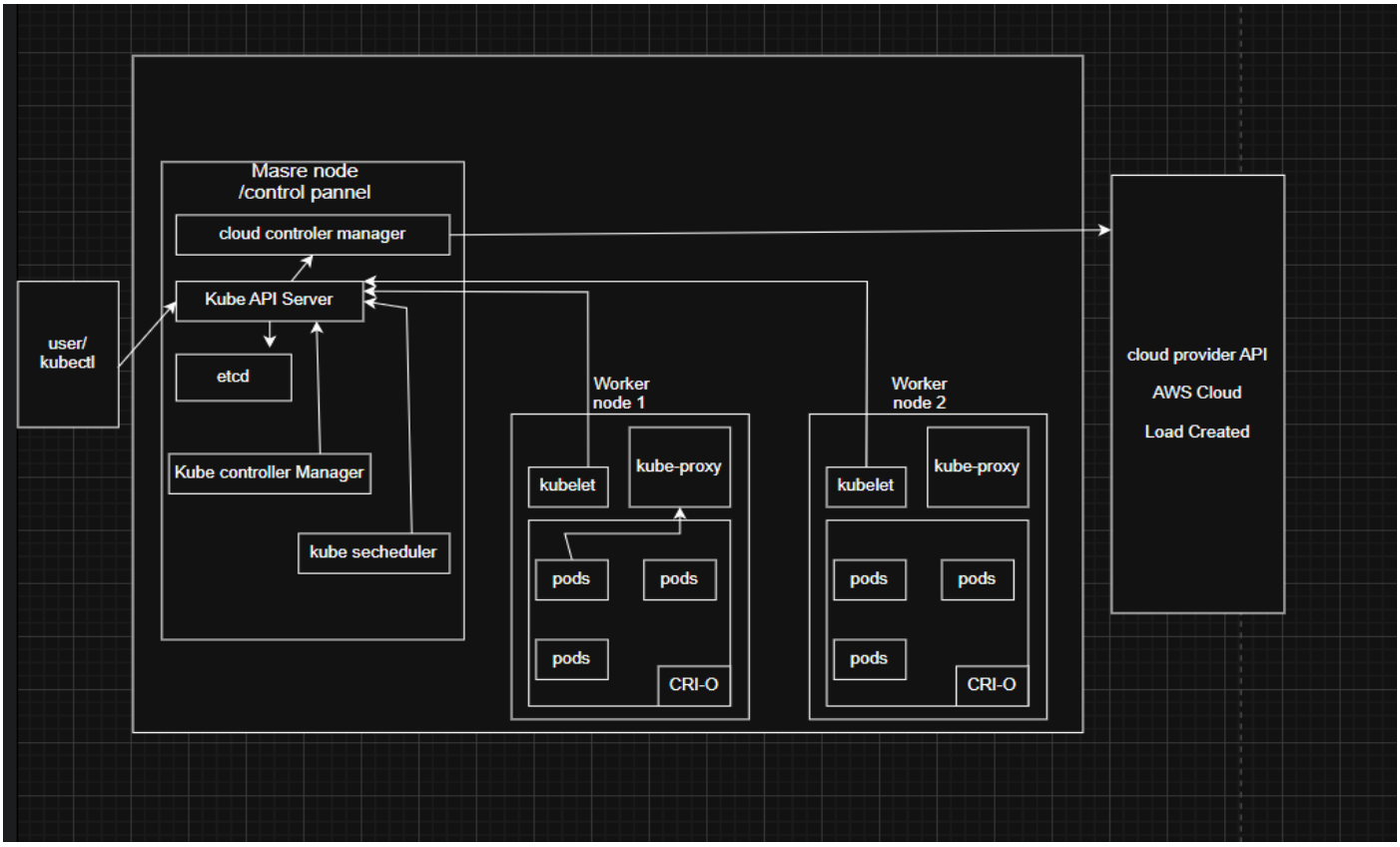
K8s document



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:-

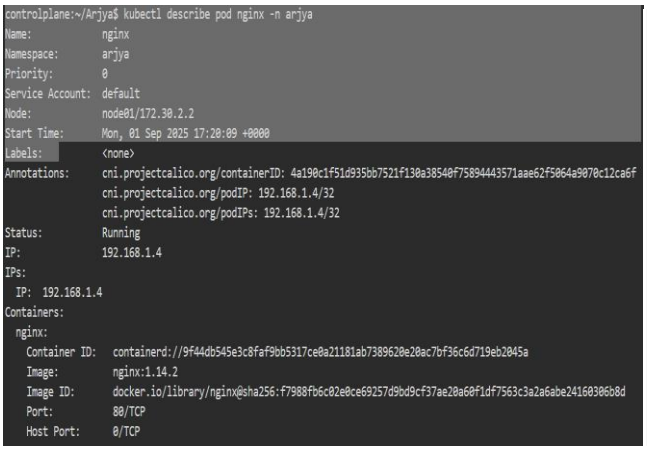
1.Kubectl apply -f pod.yaml : for creating pod

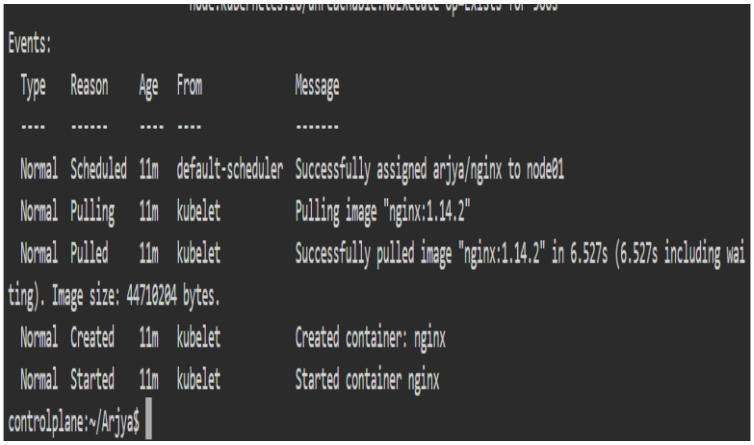
```
controlplane:~/Arjya$ kubectl apply -f pod.yaml
pod/nginx created
```

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

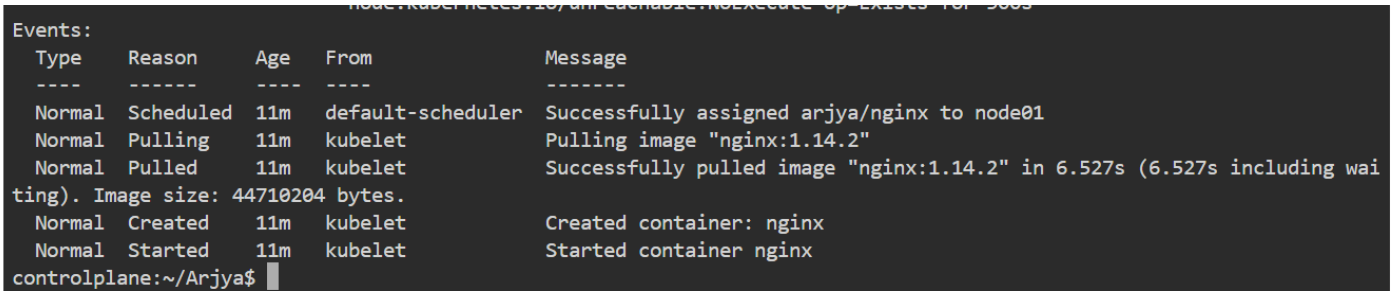
```
controlplane:~/Arjya$ kubectl get pods -n arjya
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          27s
controlplane:~/Arjya$ cat pod.yaml
```

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

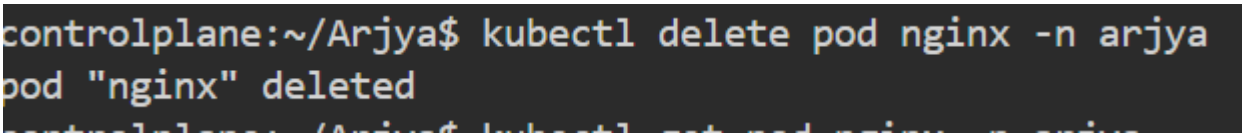




4. **kubectl exec -it nginx -n arjya -- /bin/bash** :- Execute a command inside the Pod



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



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:

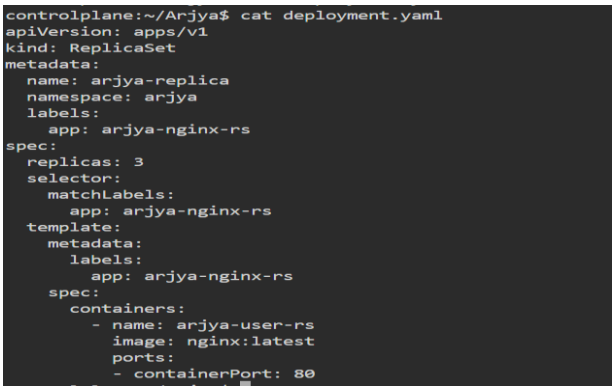
- name: arjya-user-dp

image: nginx:1.21.3

ports:

- containerPort: 80

1.



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

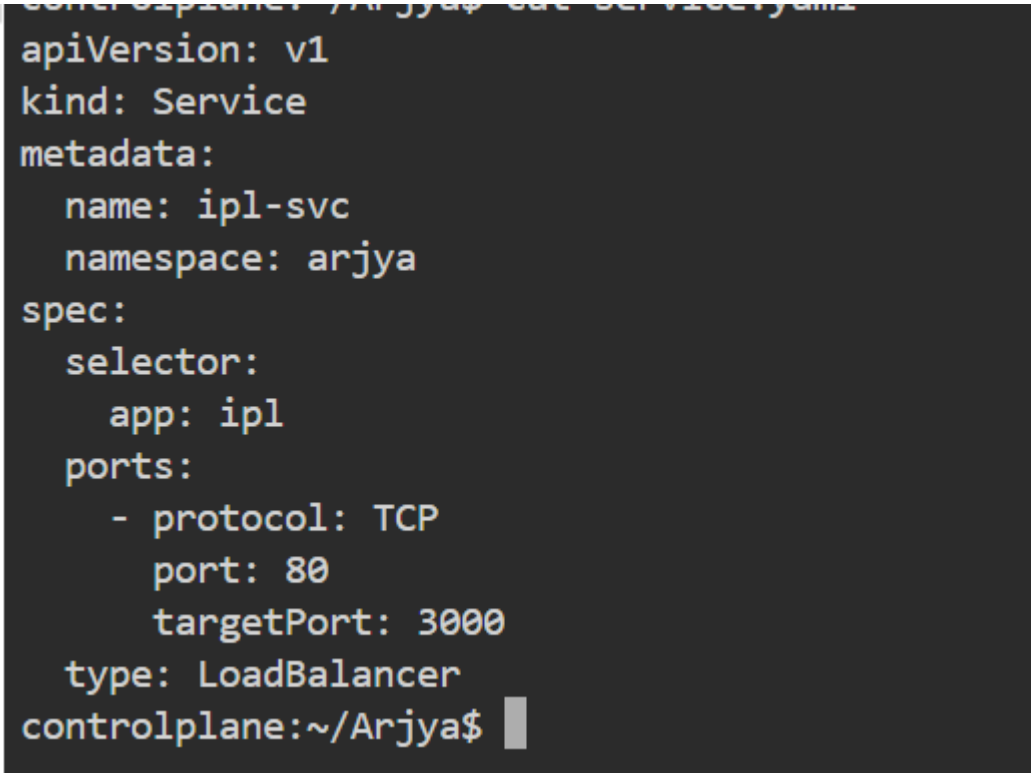
  ports:

  - protocol: TCP

    port: 80

    targetPort: 3000

  type: LoadBalancer



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$
```