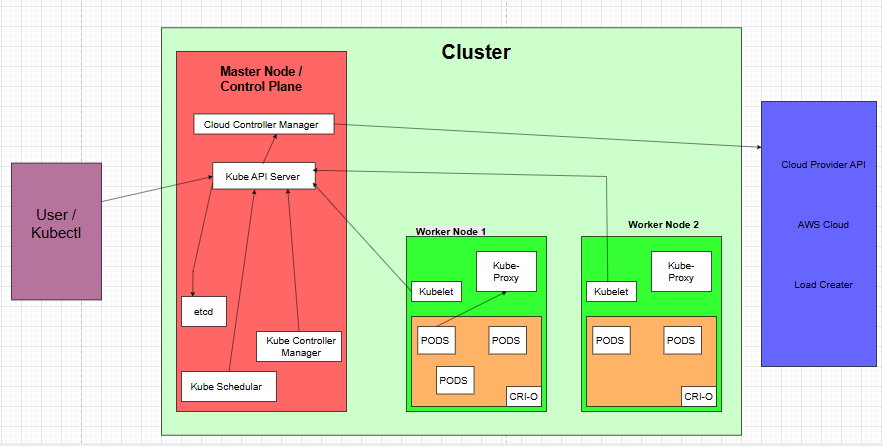
**Kubernetes**

**Kubernetes Architecture**



* **User / Kubectl:** A CLI tool that interacts with the cluster by sending commands to the Kube-API Server.
* **CONTROL PLANE (Master Node):** This is the brain of the cluster, responsible for managing the cluster state and orchestrating workloads.
* **Kube-API Server:** The front-end of the control plane. All requests (kubectl, UI, CLI, other components) go through this API. Validates and processes REST requests.
* **etcd:** A key-value store that stores all cluster data/state. Highly available and consistent. The API server reads from and writes to etcd.
* **Kube Controller Manager:** Runs controller loops (e.g., ReplicationController, NodeController, JobController) that regulate the state of the cluster.
* **Kube Scheduler:** Assigns Pods to nodes based on resource availability and scheduling policies.
* **Cloud Controller Manager:** Integrates with cloud provider APIs (e.g., AWS, Azure) to manage infrastructure like load balancers, volumes, and node lifecycle.
* **WORKER NODE:** This is where your applications (Pods) actually run.
* **Kubelet:** The agent on each worker node. Ensures that containers are running in a Pod as instructed by the control plane. Communicates with the API server.
* **Kube-Proxy:** Maintains network rules on nodes. Handles routing and load-balancing to ensure Pods can communicate across nodes.
* **Container Runtime (CRI-O):** Interface to run containers. Alternatives include containerd or Docker. It pulls images, starts/stops containers.
* **PODs:** The smallest deployable unit. A Pod encapsulates one or more containers.

**POD**

apiVersion: v1

kind: Pod

metadata:

name: pod

namespace: prince

labels: prince-app

spec:

containers:

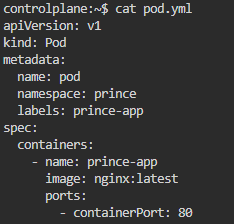
- name: prince-app

image: nginx:latest

ports:

- containerPort: 80

* **apiVersion:** This specifies the version of the Kubernetes API you are using for this configuration.
* **Kind:** This defines the type of Kubernetes object you're creating.
* **Pod:** It is the smallest deployable unit in Kubernetes that can contain one or more containers.
* **Metadata:** Metadata provides information about the object, like its name, namespace, labels, etc.
* **Namespace:** Namespaces are used to divide cluster resources between multiple users or teams.
* **Labels:** Labels are used for grouping, selecting, and filtering Kubernetes objects.
* **Spec:** This section defines the desired state and specifications of the Pod — including containers, volumes, and other configurations.
* **Containers:** A list of containers that will run inside the Pod.
* **image: nginx:latest:** The container will use the nginx image, pulling the latest version from Docker Hub.
* **Ports:** List of ports that the container exposes.



**Commands**

* **Create a Pod from YAML**

kubectl apply -f pod.yaml

* **View All Pods (in current namespace)**

kubectl get pods

* **View All Pods (in all namespaces)**

kubectl get pods --all-namespaces

* **Describe a Specific Pod**

kubectl describe pod <pod-name>

* **Check Logs of a Pod**

kubectl logs <pod-name>

* **Check Logs of a Pod’s Specific Container**

kubectl logs <pod-name> -c <container-name>

* **Execute a Command Inside a Pod (like SSH)**

kubectl exec -it <pod-name> -- /bin/bash

* **Execute a One-Time Command in a Pod**

kubectl exec <pod-name> -- ls /app

* **Delete a Pod**

kubectl delete pod <pod-name>

* **Create a Pod Quickly (Imperatively)**

kubectl run nginx-pod --image=nginx

* **Watch Pods in Real Time**

kubectl get pods –watch

* **List Pods with Specific Label**

kubectl get pods -l app=nginx

**ReplicaSet**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: prince-replicaset

namespace: prince

labels:

app: prince-rs

spec:

replicas: 2

selector:

matchLabels:

app: prince-rs

template:

metadata:

labels:

app: prince-rs

spec:

containers:

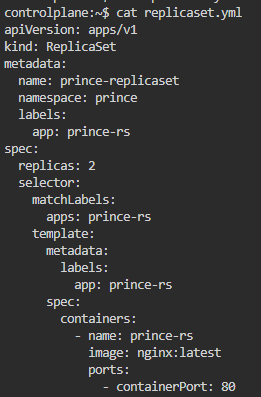
- name: prince-rs

image: nginx:latest

ports:

- containerPort: 80

* **Replicaset:** It ensures that a specified number of Pod replicas are running at all times.
* **Selector:** This defines how the ReplicaSet identifies which Pods it manages — using label matching.
* **Template:** This is a Pod template that tells the ReplicaSet what Pods to create and manage.



**Commands**

* **Create a ReplicaSet from YAML**

kubectl apply -f replicaset.yaml

* **List All ReplicaSets in Current Namespace**

kubectl get rs

* **List All ReplicaSets Across All Namespaces**

kubectl get rs --all-namespaces

* **Describe a Specific ReplicaSet**

kubectl describe rs <replicaset-name>

* **Delete a ReplicaSet**

kubectl delete rs <replicaset-name>

* **Scale a ReplicaSet (Change Number of Replicas)**

kubectl scale rs <replicaset-name> --replicas=<number>

* **Get Pods Managed by a ReplicaSet (Using Label Selector)**

kubectl get pods -l <label-selector>

* **Watch ReplicaSets in Real Time**

kubectl get rs –watch

* **Patch a ReplicaSet**

kubectl patch rs <replicaset-name> -p '{"spec":{"replicas":3}}'

* **Edit a ReplicaSet**

kubectl edit rs <replicaset-name>

* **Rollout Status of ReplicaSet**

kubectl rollout status rs/<replicaset-name>

**Deployment**

apiVersion: apps/v1

kind: Deployment

metadata:

name: prince-dp

namespace: prince

labels:

app: prince-nginx-dp

spec:

replicas: 2

selector:

matchLabels:

app: prince-dp

template:

metadata:

labels:

app: prince-dp

spec:

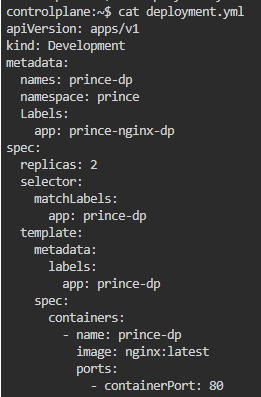
containers:

- name: prince-dp

image: nginx:latest

ports:

- containerPort: 80



* **Create a Deployment from YAML**

kubectl apply -f deployment.yaml

* **List All Deployments in Current Namespace**

kubectl get deployments

* **List All Deployments Across All Namespaces**

kubectl get deployments --all-namespaces

* **Describe a Specific Deployment**

kubectl describe deployment <deployment-name>

* **Delete a Deployment**

kubectl delete deployment <deployment-name>

* **Scale a Deployment (Change Number of Replicas)**

kubectl scale deployment <deployment-name> --replicas=<number>

* **Watch Deployments in Real Time**

kubectl get deployments –watch

* **Edit a Deployment**

kubectl edit deployment <deployment-name>

* **Patch a Deployment**

kubectl patch deployment <deployment-name> -p '{"spec":{"replicas":3}}'

* **Rollout a Deployment Update**

kubectl rollout restart deployment <deployment-name>

* **Check Rollout Status**

kubectl rollout status deployment/<deployment-name>

* **Undo Last Rollout**

kubectl rollout undo deployment/<deployment-name>

* **Get Deployment Rollout History**

kubectl rollout history deployment/<deployment-name>

* **Pause a Deployment**

kubectl rollout pause deployment/<deployment-name>

* **Resume a Deployment**

kubectl rollout resume deployment/<deployment-name>

**Cluster IP Services**

apiVersion: v1

kind: Service

metadata:

name: clusterIP-service

spec:

selector:

app: clusterIP

ports:

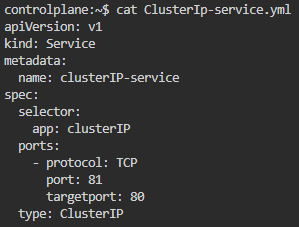
- protocol: TCP

port: 81

targetPort: 80

type: ClusterIP

* **Service:** A Kubernetes Service exposes a set of Pods as a network service. It enables communication inside the cluster (or outside, depending on type).
* **Type:** This defines the type of service being created.
* **ClusterIP:** ClusterIP exposes the service internally within the cluster only — not accessible from outside.
* **Protocol:** Specifies the network protocol (default is TCP).



**Commands**

* **Create a Service from YAML**

kubectl apply -f service.yaml

* **List All Services in Current Namespace**

kubectl get svc

* **List All Services Across All Namespaces**

kubectl get svc --all-namespaces

* **Describe a Specific Service**

kubectl describe svc <service-name>

* **Delete a Service**

kubectl delete svc <service-name>

* **Get Service YAML Definition**

kubectl get svc <service-name> -o yaml

* **Watch Services in Real Time**

kubectl get svc –watch

* **List Services with a Specific Label**

kubectl get svc -l <label-key>=<label-value>

* **Port Forward to a Service**

kubectl port-forward svc/<service-name> <local-port>:<service-port>

* **Check Endpoints of a Service**

kubectl get endpoints <service-name>

* **Edit a Service**

kubectl edit svc <service-name>

* **Patch a Service**

kubectl patch svc <service-name> -p '{"spec":{"ports":[{"port":8080}]}}'

* **Expose a Deployment as a Service**

kubectl expose deployment <deployment-name> --port=80 --target-port=8080 --name=<service-name> --type=ClusterIP

**Nodeport Services**

apiVersion: v1

kind: Service

metadata:

name: nodePort-service

spec:

selector:

app: prince

ports:

- protocol: TCP

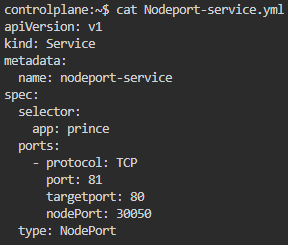
port: 81

targetPort: 80

nodePort: 30050

type: NodePort

* **NodePort:** This exposes the service outside the cluster using a port on each node's IP address.
* **port:** The port that other services in the cluster use to reach this service.
* **targetPort:** The port on the Pod/container that receives the traffic.
* **nodePort:** The port on the Node's IP that external users can use to access the service.



**Load Balancer Services**

apiVersion: v1

kind: Service

metadata:

name: loadBalancer-service

spec:

selector:

app: prince

ports:

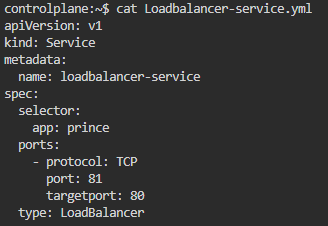
- protocol: TCP

port: 81

targetPort: 80

type: LoadBalancer

* **LoadBalancer:** A Load Balancer is a component that distributes incoming network traffic across multiple servers.



External Name

apiVersion: v1

kind: Service

metadata:

name: externalName-service

namespace: prince

spec:

selector:

app: prince

ports:

- protocol: TCP

port: 81

externalName: google.com

type: ExternalName

* **ExternalName:** Used to map a Kubernetes service name to an external DNS name

