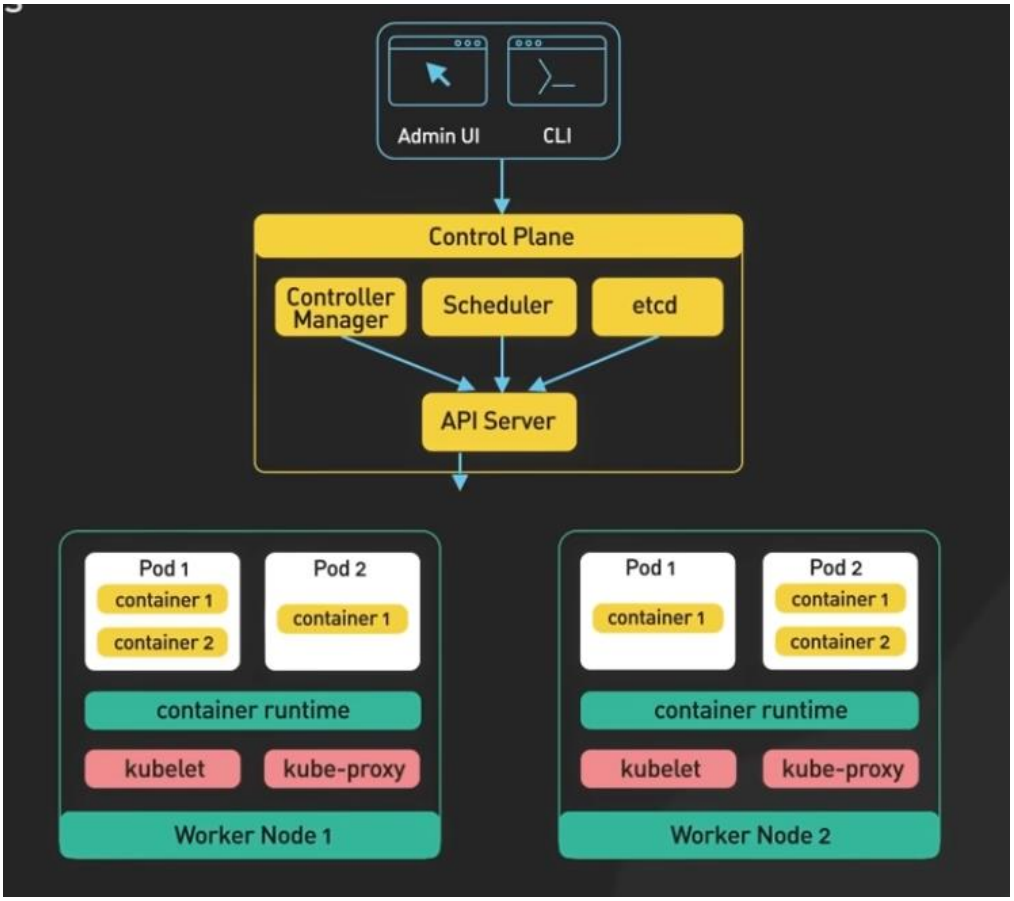
Control Plane:



1. API Server (`kube-apiserver`)

- **Role**: Frontend REST API for cluster operations.
- **Key Commands**:
  - Check status:
    `kubectl get componentstatuses` (`kubectl get cs`)
  - View logs (systemd):
    `journalctl -u kube-apiserver`
  - Health endpoint:
    `curl -k https://<API_SERVER_IP>:6443/healthz`

2. etcd

- **Role**: Distributed key-value store for cluster state.
- **Key Commands**:
  - Check cluster health:
    `etcdctl endpoint health --endpoints=<ETCD_ENDPOINTS>`
  - List members:
    `etcdctl member list`
  - Snapshot backup:
    `etcdctl snapshot save /path/to/snapshot.db`

- Restore snapshot:
  `etcdctl snapshot restore /path/to/snapshot.db`

3. Scheduler (`kube-scheduler`)

- **Role**: Assigns pods to nodes.
- **Key Commands**:
  - View logs:
    `journalctl -u kube-scheduler`
  - Check scheduling decisions:
    `kubectl describe pod <POD_NAME> | grep Events`

4. Controller Manager (`kube-controller-manager`)

- **Role**: Runs core controllers (node, replication, endpoints).
- **Key Commands**:
  - View logs:
    `journalctl -u kube-controller-manager`

6. Kubelet

- **Role**: Runs on nodes, manages pods and containers.
- **Key Commands**:
  - Check status:
    `systemctl status kubelet`
  - View logs:
    `journalctl -u kubelet`

7. Kubectl (CLI Tool)

- **Common Commands:**
  - **Get resources:** `kubectl get <pods/nodes/services>`
  - **Describe resource:** `kubectl describe <pod/node> <NAME>`
  - **View logs:** `kubectl logs <POD_NAME>`
  - **Exec into pod:** `kubectl exec -it <POD_NAME> -- /bin/sh`

8. kube-proxy

it is responsible for managing network rules to ensure communication between **services**, **pods**, and external clients works correctly.

Key Commands

- **Check Service Endpoints**:
  kubectl get endpoints <SERVICE_NAME>
- **Check kube-proxy config**:
  kubectl describe configmap -n kube-system kube-proxy

11. Namespaces

- **`default`**: Created automatically.
- **`kube-system`**: Hosts Control Plane components (e.g., `kube-proxy`, `CoreDNS`).
- **`kube-public`**: Readable by all users (rarely used).
- **`kube-node-lease`**: Node heartbeat tracking.

Namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
 name: dev
---
apiVersion: v1
kind: Namespace
metadata:
 name: prod
```

12. Pod

```
 apiVersion: v1
kind: Pod
metadata:
 name: advanced-pod
 labels:
   app: nginx
namespace: production
   tier: frontend
spec:
 # --- Containers Section ---
 containers:
 - name: nginx
   image: nginx:1.25-alpine
   imagePullPolicy: IfNotPresent  # Options: Always/Never/IfNotPresent

   # --- Ports ---
   ports:
   - containerPort: 80
     protocol: TCP
     name: http

   # --- Resource Limits ---
   resources:
     requests:
       cpu: "250m"    # 0.25 CPU cores
       memory: "256Mi" # 256 MB RAM
     limits:
```

```
      cpu: "1"       # 1 CPU core max
      memory: "512Mi" # 512 MB RAM max

  # --- Liveness Probe ---
  livenessProbe:
    httpGet:
      path: /healthz
      port: 80
    initialDelaySeconds: 15  # Wait 15s before first probe
    periodSeconds: 10        # Check every 10s
    timeoutSeconds: 2        # Probe timeout
    failureThreshold: 3      # 3 failures = container restart

  # --- Readiness Probe ---
  readinessProbe:
    tcpSocket:
      port: 80
    initialDelaySeconds: 5
    periodSeconds: 5
    successThreshold: 1
    failureThreshold: 3

  # --- Environment Variables ---
  env:
  - name: NGINX_ENV
    value: "production"
  - name: DB_HOST
    valueFrom:
      secretKeyRef:
        name: db-secret
        key: host

  # --- Volume Mounts ---
  volumeMounts:
  - name: config-volume
    mountPath: /etc/nginx/conf.d
    readOnly: true
  - name: logs-volume
    mountPath: /var/log/nginx

  # --- Security Context ---
  securityContext:
    runAsNonRoot: true
    runAsUser: 1000
runAsGroup: 3000
```

```
    fsGroup: 2000
    readOnlyRootFilesystem: true
    capabilities:
      drop: ["ALL"]
      add: ["NET_BIND_SERVICE"]

 # --- Init Containers ---
 initContainers:
 - name: init-db
   image: busybox:1.28
   command: ['sh', '-c', 'until nslookup db-service; do echo waiting for db; sleep 2; done']

 # --- Volumes ---
 volumes:
 - name: config-volume
   configMap:
     name: nginx-config
 - name: logs-volume
   emptyDir: {}

 # --- Pod-wide Settings ---
 restartPolicy: Always  # Options: Always/OnFailure/Never
 nodeSelector:
   disktype: ssd       # Schedule on nodes with this label
 tolerations:
 - key: "special"
   operator: "Exists"
   effect: "NoSchedule"
 securityContext:
   fsGroup: 2000       # File ownership group

Deployments
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
   app: nginx
   tier: frontend
spec:
 replicas: 3  # Number of Pods to maintain
 selector:
   matchLabels:
     app: nginx
     tier: frontend
```

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 1       # How many extra Pods can be created during update
    maxUnavailable: 0 # No Pods should be unavailable during update
template:
  metadata:
    labels:
      app: nginx
      tier: frontend
  spec:
    # --- Containers Section ---
    containers:
    - name: nginx
      image: nginx:1.25-alpine
      imagePullPolicy: IfNotPresent
      ports:
      - containerPort: 80
        protocol: TCP
        name: http
      resources:
        requests:
          cpu: "250m"
          memory: "256Mi"
        limits:
          cpu: "1"
          memory: "512Mi"
      livenessProbe:
        httpGet:
          path: /healthz
          port: 80
        initialDelaySeconds: 15
        periodSeconds: 10
        timeoutSeconds: 2
        failureThreshold: 3
      readinessProbe:
        tcpSocket:
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 5
        successThreshold: 1
        failureThreshold: 3
      env:
      - name: NGINX_ENV
        value: "production"
```

```
      - name: DB_HOST
        valueFrom:
          secretKeyRef:
            name: db-secret
            key: host
      volumeMounts:
      - name: config-volume
        mountPath: /etc/nginx/conf.d
        readOnly: true
      - name: logs-volume
        mountPath: /var/log/nginx
      securityContext:
        runAsNonRoot: true
        runAsUser: 1000
        readOnlyRootFilesystem: true
        capabilities:
          drop: ["ALL"]
          add: ["NET_BIND_SERVICE"]

    # --- Init Containers ---
    initContainers:
    - name: init-db
      image: busybox:1.28
      command: ['sh', '-c', 'until nslookup db-service; do echo waiting for db; sleep 2; done']

    # --- Volumes ---
    volumes:
    - name: config-volume
      configMap:
        name: nginx-config
    - name: logs-volume
      emptyDir: {}

    # --- Pod-wide Settings ---
    nodeSelector:
      disktype: ssd
    tolerations:
    - key: "special"
      operator: "Exists"
      effect: "NoSchedule"
    securityContext:
      fsGroup: 2000
Replica set
apiVersion: apps/v1
kind: ReplicaSet
```

```
metadata:
 name: frontend-rs
 labels:
   app: frontend
   tier: web
spec:
 replicas: 3
 selector:
   matchLabels:
     tier: web  # Must match pod template labels
 template:
   metadata:
     labels:
       tier: web  # Must match selector
   spec:
     containers:
     - name: nginx
       image: nginx:1.25
       ports:
       - containerPort: 80
```

SERVICES

1. ClusterIP (Internal Service)

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
 labels:
   app: nginx
   tier: frontend
spec:
 type: ClusterIP  # Default - accessible only within the cluster
 selector:
   app: nginx     # Must match Deployment's pod labels
   tier: frontend
 ports:
 - name: http
   port: 80       # Service port
   targetPort: 80 # Pod port (matches containerPort)
   protocol: TCP
```

2. NodePort (External Access via Node IP)

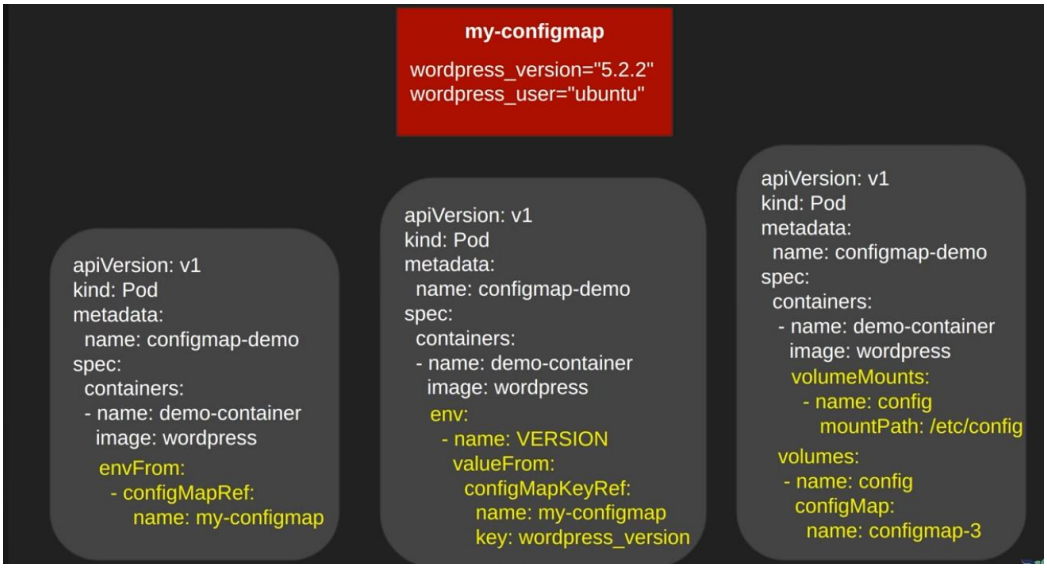
```
apiVersion: v1
kind: Service
metadata:
```

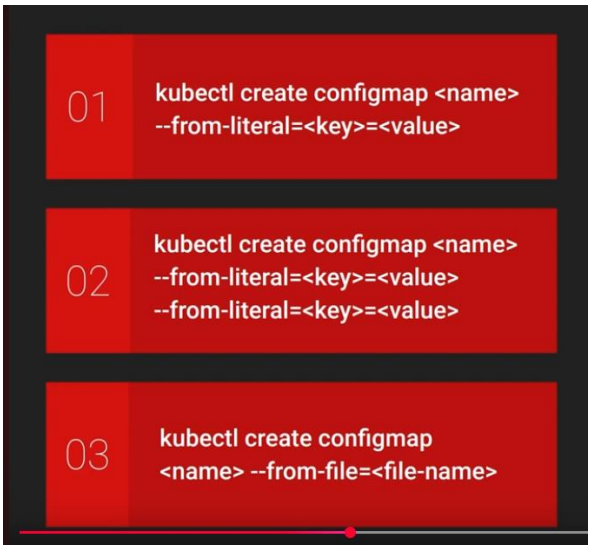
```
 name: nginx-nodeport
spec:
 type: NodePort
 selector:
   app: nginx
   tier: frontend
 ports:
 - port: 80
   targetPort: 80
   nodePort: 30080  # Optional (default range: 30000-32767)
```

3. LoadBalancer (Cloud Provider Integration)

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-lb
 annotations:
   cloud-provider-specific-key: "value"  # e.g., AWS: service.beta.kubernetes.io/aws-load-
balancer-type: nlb
spec:
 type: LoadBalancer
 selector:
   app: nginx
   tier: frontend
 ports:
 - port: 80
   targetPort: 80
 externalTrafficPolicy: Local  # Preserves client IP
```

Config Map





Secrets

Generic Secret

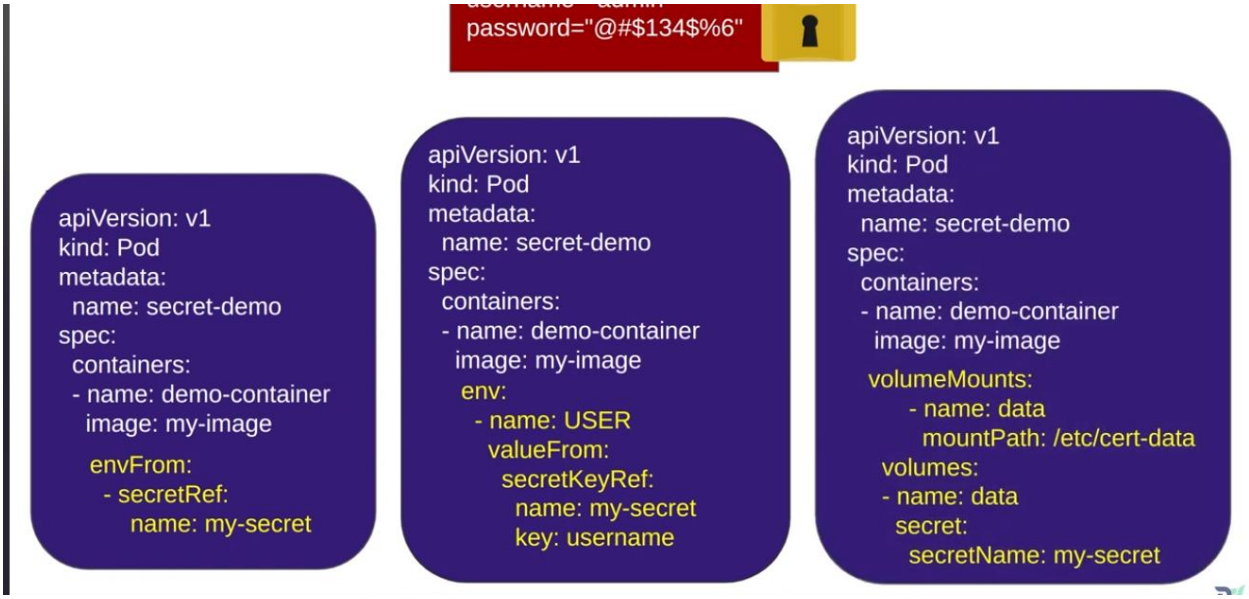
kubectl create secret generic db-secret --from-literal=username=dbuser --from-literal=password=Y4nys7f11

Docker-registry Secret

kubectl create secret docker-registry docker-secret --docker-email=example@gmail.com --docker-username=dev --docker-password=pass1234 --docker-server=my-registry.example:5000

TLS Secret

kubectl create secret tls my-tls-secret --cert=/root/data/serverca.crt --key=/root/data/servercakey.pem



Volumes, PV,PVC

Workflow

1. **Admin** creates StorageClass (defines storage types).
2. **User** creates pvc (requests storage).
3. **Kubernetes**:
    - Dynamically provisions PV (cloud) **or**
    - Binds to existing PV (on-prem)
4. **Pod** mounts PVC → accesses persistent storage.

Storage class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
parameters:
  type: gp2
reclaimPolicy: Delete  # Or 'Retain'
volumeBindingMode: WaitForFirstConsumer
```

pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ebs-claim
spec:
  accessModes:
        - ReadWriteOnce  # RWO, ROX, RWX
```

```
  storageClassName: ebs-sc
  resources:
        requests:
        storage: 1Gi
```

pod.yaml

```
//Baki file of pod
        volumeMounts:
        - name: persistent-storage
        mountPath: /data
  volumes:
  - name: persistent-storage
        persistentVolumeClaim:
        claimName: ebs-claim
```

Service Account

Role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - watch
  - list
```

RoleBinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: ServiceAccount
  name: mysa
  namespace: default
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

In pod.yaml

```
spec:
```

```
  serviceAccountName: mysa
```
Verifying Permissions => kubectl auth can-i get pods --as=system:serviceaccount:default:mysa

HELM

```
my-chart/
├── Chart.yaml          # Chart metadata
├── values.yaml         # Default configuration
├── charts/             # Sub-charts
├── templates/          # Kubernetes manifests
│   ├── deployment.yaml
│   ├── service.yaml
│   └── _helpers.tpl    # Reusable templates
└── templates/NOTES.txt # Post-install notes
```

templates/deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: {{ .Release.Name }}-app  # Built-in object
spec:
 replicas: {{ .Values.replicaCount }}  # Value reference
 template:
   spec:
     containers:
     - name: {{ .Chart.Name }}  # Built-in
       image: "{{ .Values.image.repo }}:{{ .Values.image.tag }}"
       # --- IF/ELSE EXAMPLE ---
       {{- if .Values.resources }}
       resources: {{ toYaml .Values.resources | nindent 10 }}
       {{- else }}
       resources: {}
       {{- end }}
       # --- RANGE EXAMPLE ---
       env:
       {{- range $key, $val := .Values.env }}
       - name: {{ $key | upper }}
         value: {{ $val | quote }}
       {{- end }}
       # --- WITH SCOPE EXAMPLE ---
       {{- with .Values.securityContext }}
       securityContext: {{ . | toYaml | nindent 10 }}
       {{- end }}
```

values.yaml

```
# --- BASIC VALUES ---
replicaCount: 2

# --- NESTED VALUES ---
image:
 repo: nginx
 tag: latest

# --- IF/ELSE TARGET ---
resources:
 limits:
   cpu: "500m"
   memory: "512Mi"

# --- RANGE TARGET ---
env:
 log_level: debug
 cache_enabled: "true"

# --- WITH SCOPE TARGET ---
securityContext:
 runAsNonRoot: true



{{ .Values.text | quote }}          # Add quotes
{{ .Values.data | toYaml | indent 4 }}  # YAML+indent


# Install with overrides
helm install my-app . --set replicaCount=3

# Dry-run template rendering
helm template . --set image.tag=canary

Named Template
_helpers.tpl
{{- define "labels" }}
          app: nginx
          env: dev
{{- end }}
Deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
 name: nginx-deployment
 labels:
   {{- template "labels" }}
spec:
 replicas: 1
 selector:
   matchLabels:
     {{- include "labels" . | indent 2 }}
 template:
   metadata:
     labels:
       {{- include "labels" . | indent 4 }}
   spec:
     containers:
     - name: nginx
       image: nginx:1.14.2
       ports:
       - containerPort: 80
```