Day-14 SRE Training

Topic: Kubernetes

Why Kubernetes?

Docker provides containerization, but Kubernetes provides **orchestration** of containers. This means:

1. **Automated Deployment**: Deploy containers across multiple hosts.
2. **Scaling**: Scale containers up or down based on demand.
3. **Self-healing**: Restart or replace failed containers automatically.
4. **Service Discovery**: Find and communicate with services dynamically.
5. **Load Balancing**: Distribute traffic across container instances.
6. **Rolling Updates**: Update applications without downtime.

Core Kubernetes Components

Pod

- Smallest deployable unit in Kubernetes.
- Contains one or more containers that share storage and network.
- Usually one main application container per pod.
- Think of it as a logical host for your container(s).

Deployment

- Manages a set of identical pods (replicas).
- Ensures the specified number of pods are running.
- Handles rolling updates and rollbacks.
- Maintains pod health and replaces failed pods.

Service

- Provides a stable network endpoint to access pods.
- Pods are **ephemeral** (temporary), but services are **persistent**.
- **Types of Services:**
    - **ClusterIP**: Internal only.
    - **NodePort**: Exposes on Node IP at a static port.
    - **LoadBalancer**: Exposes externally using a cloud provider's load balancer.

ConfigMap & Secret

- **ConfigMap**: Stores **non-sensitive** configuration.
- **Secret**: Stores **sensitive** data (passwords, tokens, keys).
- Both **decouple configuration** from container images.

Kubernetes Architecture

Control Plane Components

- **API Server**: Front-end to the control plane; all communication goes through it.
- **etcd**: Key-value store that holds all cluster data.
- **Scheduler**: Assigns pods to nodes.
- **Controller Manager**: Runs controller processes (e.g., deployment controller).

Node Components

- **kubelet**: Agent that ensures containers are running in a pod.
- **kube-proxy**: Maintains network rules on nodes.
- **Container Runtime**: Software responsible for running containers (e.g., Docker).

Kubernetes vs Docker

Feature	Docker	Kubernetes	Example
Focus	Creating and running containers	Orchestrating containers at scale	Docker runs a single container for a personal project, while Kubernetes manages 100s of containers for a cloud app.
Scale	Single host or small-scale	Multi-host clusters	Docker is great for local development, while Kubernetes is used for multi-server deployment of microservices.
Self-healing	Limited	Automatic pod replacement	If a container crashes in Docker, manual restart is needed, but Kubernetes auto-replaces failed pods.
Load Balancing	Basic	Advanced internal and external	Docker needs manual load balancing, while Kubernetes distributes traffic automatically across pods.
Scaling	Manual	Automatic horizontal scaling	In Docker, you must manually increase containers, but Kubernetes scales up/down based on demand.
Updates	Manual	Automated rolling updates	Updating a Docker app requires downtime, whereas Kubernetes does rolling updates with zero downtime.

These YAML files define different Kubernetes resources:

1. **namespace.yaml** – Creates a logical separation (namespace) in a Kubernetes cluster to group resources.
2. **configmap.yaml** – Stores non-sensitive configuration data (e.g., environment variables) that can be used by pods.
3. **deployment.yaml** – Defines how to deploy and manage application pods, including replicas, rolling updates, and restarts.
4. **service.yaml** – Exposes deployed pods internally (ClusterIP) or externally (NodePort, LoadBalancer) for communication

kubectl – A command-line tool to interact with the Kubernetes cluster (e.g., deploy apps, manage pods, view logs).

View running pods: kubectl get pods

View services: kubectl get svc

Check logs: kubectl logs <pod-name>

kubectl get deployments -n <namespace>: Lists all deployments in the specified namespace.

kubectl get namespace: Displays all namespaces in the Kubernetes cluster.
kubectl get pods -n <namespace>:Lists all pods running in the specified namespace.

Minikube – A lightweight Kubernetes cluster that runs locally, useful for testing and development.
View url of the running application : minikube service <service-name> --url

---

Kubernetes Mini Demo

App: Kubernetes Mini Demo v1.0.0

Hostname (Pod name): flask-app-855886647-bm5wt

Pod IP: 10.244.0.5

Request count: 1

Time: 2025-02-27 05:39:41

View API Info

Health Check