**SRE TRAINING (DAY 14) - KUBERNETES**

**What is Kubernetes?**

**Kubernetes (K8s)** is an **open-source** platform for **automating deployment, scaling, and management** of containerized applications.

Before Kubernetes, applications were deployed on virtual machines (VMs) or bare-metal servers. Containers (like Docker) improved efficiency, but managing multiple containers across different machines manually became complex.

Kubernetes solves this problem by:

✅ Automating container deployment & scaling.

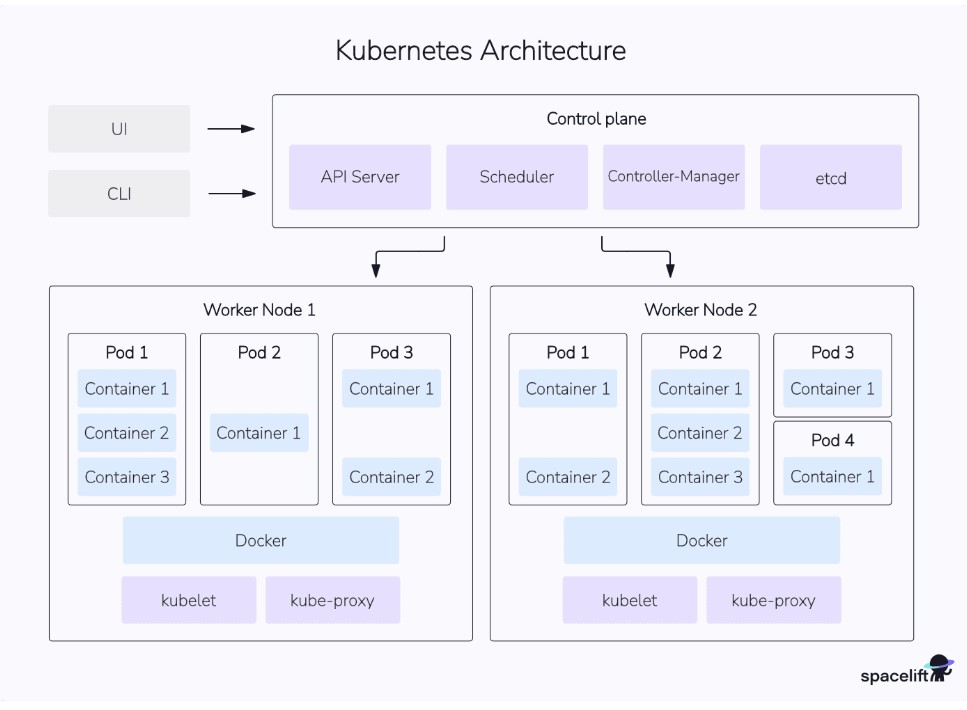
✅ Managing networking & storage for containers.

✅ Ensuring high availability and fault tolerance.

✅ Rolling out updates & rollbacks easily

**Kubernetes**

**Architecture**



**1**⃣ **Control Plane (Master Node)**

**The brain of Kubernetes that manages the cluster**. It includes:

* **API Server**: The entry point for all Kubernetes commands (kubectl, Minikube, etc.).
* **Scheduler**: Assigns workloads (Pods) to worker nodes based on resource availability.
* **Controller Manager**: Monitors the cluster and ensures desired states are maintained.
* **etcd**: A distributed key-value store that keeps cluster state.

**2**⃣ **Worker Nodes**

These are the **machines where your applications run**. Each node contains:

* **Kubelet**: A small agent that ensures Pods are running.
* **Container Runtime**: Runs the actual containers (e.g., Docker).
* **Kube Proxy**: Handles networking within the cluster.

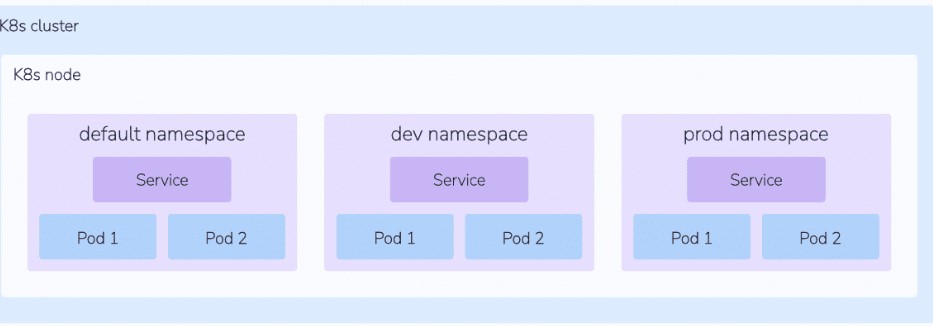
# Basic terms and concepts

## 1. Kubernetes Nodes

**Nodes** represent the physical machines that form your [Kubernetes cluster](https://spacelift.io/blog/kubernetes-cluster). They run the containers you create. Kubernetes tracks the status of your nodes and exposes each one as an object.

## 2. Namespaces

[Kubernetes **namespaces**](https://spacelift.io/blog/kubernetes-namespaces) isolate different groups of resources. They avoid name collisions by scoping the visibility of your resources.



## 3. Pods

[Pods](https://kubernetes.io/docs/concepts/workloads/pods) are the fundamental compute unit in Kubernetes. A Pod is analogous to a container but with some key differences. Pods can contain multiple containers, each of which share a context. The entire Pod will always be scheduled onto the same node. The containers within a Pod are tightly coupled so you should create a new Pod for each distinct part of your application, such as its API and database.

## 4. Minikube

**Minikube** is a **lightweight Kubernetes tool** that lets you run Kubernetes on a single machine (your laptop or VM). It's used for **local development and testing** without needing a full cloud-based Kubernetes cluster.

## 5. Helm (Kubernetes Package Manager)

**Helm** is like a package manager (**apt, yum, npm**) but for **Kubernetes applications**. It simplifies installing, updating, and managing complex applications inside Kubernetes.

# How Kubernetes Works Together

1⃣ **Developer creates a Pod or Deployment** (via YAML or kubectl).

2⃣ **Kubernetes schedules it** to a **Worker Node**.

3⃣ **Kubelet on the Node** runs the container using Docker/containerd.

4⃣ **Kube Proxy** ensures networking works inside the cluster.

5⃣ **Services expose the application** (internally or externally).

6⃣ **Helm** helps manage large-scale apps easily.

# INSTALLATION

1. **Installing Kubernetes Tools (kubectl, Minikube, Docker, Helm)** 
   1. **Updated the system (apt-get update)**

**○ Installed dependencies (apt-get install -y apt-transport-https ca-certificates curl software-properties-common gnupg2 conntrack)**

**○ Installed Docker, added the current user to the docker group, and verified it with docker run hello-world.**

**○ Downloaded kubectl and moved it to /usr/local/bin/ so it could be used globally.**

**○ Installed Minikube and Helm for Kubernetes.**

**○ Configured ~/.bashrc with Kubernetes aliases.**

1. **Installing Docker** 
   1. **Any old Docker installation was removed to avoid conflicts.**

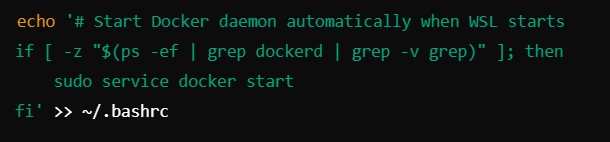
**○ The official Docker repository was added.**

**○ Docker was installed along with its dependencies.**

**○ The user was added to the docker group for permission to run Docker without sudo.**

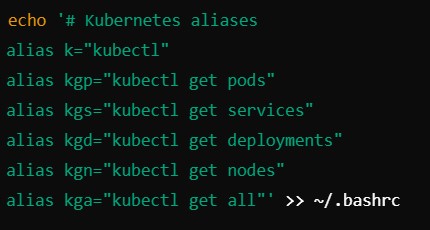
**○ A test container (hello-world) was executed to verify the installation.**

1. **Configured Docker to Start on WSL Startup**



1. **Added Kubernetes Command Aliases**

Aliases were set up to simplify frequently used Kubernetes commands.



**RUNNING THE FIRST APPLICATION**

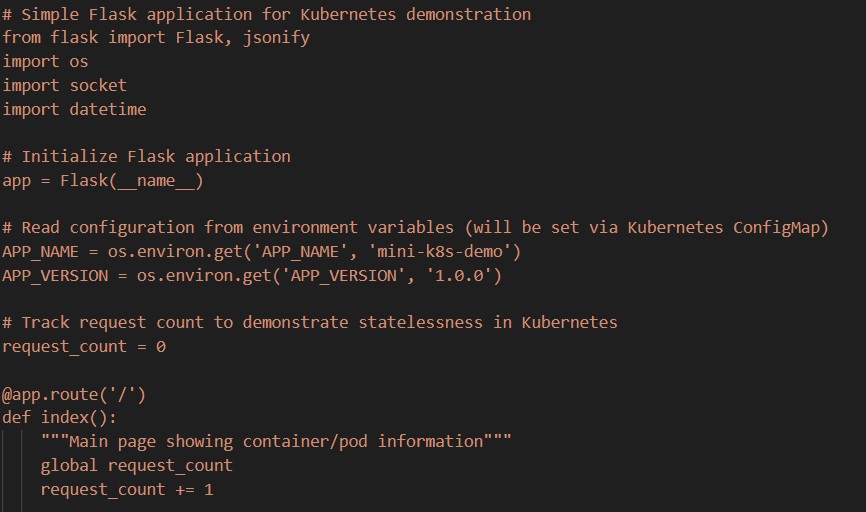
**For this we ran a kubernetes script. This script creates and deploys a minimal Kubernetes application using Flask, Docker, and Minikube.**

## Key Steps in the Script

1. **Setting Up the Project Structure** 
   * **Created a directory: ~/mini-k8s-demo/{app,k8s}**
   * **Organized files into:** 
     1. **app/ → Application code**

**○ k8s/ → Kubernetes deployment files**

1. **Creating a Flask Web Application** 
   * **Developed a Python Flask application (app.py)**
   * **Created a requirements.txt file with Flask dependency.**

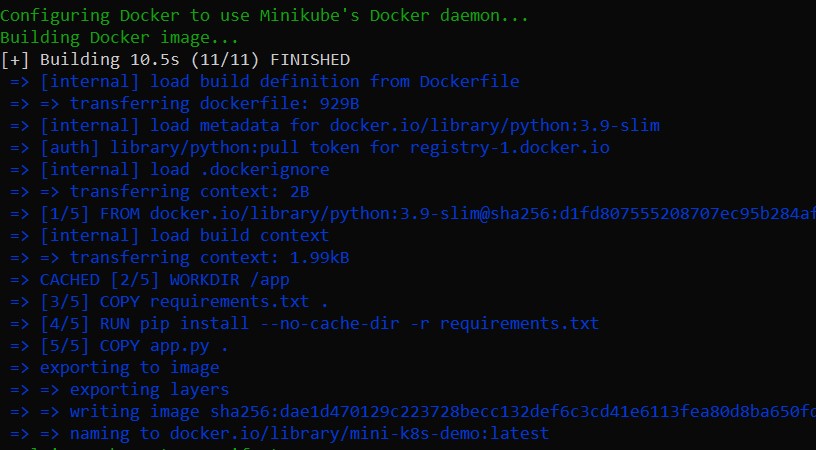


1. **Creating a Docker Image for the Application** 
   * **Created a Dockerfile with:** 
     1. **Python 3.9 slim as the base image**

**○ Installed dependencies (pip install -r requirements.txt)**

**○ Exposed port 5000**

**○ Set the default command to run Flask (CMD ["python", "app.py"])**

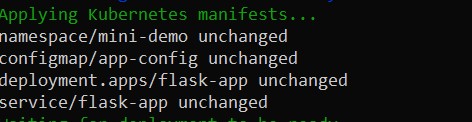


1. **Creating Kubernetes Deployment Files** 
   * **Namespace (namespace.yaml)** 
     1. **Created a new namespace mini-demo to isolate the application.**
   * **ConfigMap (configmap.yaml)** 
     1. **Stored environment variables APP\_NAME and APP\_VERSION.**
   * **Deployment (deployment.yaml)** 
     1. **Created a ReplicaSet with 2 Pods using mini-k8s-demo:latest image.**

**○ Set resource limits (cpu: 200m, memory: 128Mi).**

**○ Defined liveness and readiness probes (/api/health).**

* + **Service (service.yaml)** 
    1. **Created a NodePort service to expose the app at http://<minikube-ip>:30080.**



1. **Automating Deployment with a Shell Script (deploy.sh)** 
   * **Checked if Minikube was running and started it if needed.**
   * **Configured Docker to use Minikube's internal registry. ● Built the Docker image (mini-k8s-demo:latest).**
   * **Applied all Kubernetes manifests (kubectl apply -f k8s/).**
   * **Waited for deployment to complete (kubectl rollout status).**
   * **Printed the Minikube service URL for accessing the app.**

