DevOps with Kubernetes – Tool & Command Guide

🛠 Tools Used in the Workshop

1. Docker

- **Simple analogy:**
  Think of Docker as a **lunchbox** 🍱. You can put your food (application + everything it needs) inside, close the lid, and carry it anywhere. It will taste the same everywhere.
- **Technical detail:**
  Docker is a **containerization platform**.
  - **Containerization**: packaging software with its dependencies (libraries, configs, OS files) so it runs the same everywhere.
  - Containers are **lightweight** (share the host kernel, unlike VMs).

---

2. kubectl

- **Simple analogy:**
  kubectl is like the **remote control** 🎮 for your Kubernetes cluster. You press buttons (commands) and the cluster does what you ask (create apps, show nodes, delete things).
- **Technical detail:**
  kubectl is the **Kubernetes CLI tool**. It talks to the **Kubernetes API Server**.
  - **API Server**: central brain of Kubernetes, accepts instructions.
  - Commands like kubectl apply send **YAML manifests** (declarative config files).

---

3. kind (Kubernetes in Docker)

- **Simple analogy:**
  Imagine you want a **mini-city for practice** 🏙️ before managing a real one. kind builds a small "practice Kubernetes city" inside Docker containers on your laptop.
- **Technical detail:**
  kind = **Kubernetes IN Docker**.
  - It spins up Kubernetes clusters using **Docker containers as nodes**.
  - Perfect for **local development/testing** (not production).

---

4. Helm

- **Simple analogy:**
  Helm is like the **app store** 📲 for Kubernetes. Instead of installing apps piece by piece, you pick a Helm chart and install everything at once.

- **Technical detail:**
  Helm is a **package manager** for Kubernetes.
    - **Charts**: templates of Kubernetes YAML files packaged together.
    - You can install complex apps (Prometheus, ArgoCD, etc.) with one command.

---

5. Node.js + npm

- **Simple analogy:**
  Node.js is like a **chef that can cook with JavaScript** 🍳. Normally JS cooks only in the browser, but Node lets it cook on servers.
  npm is the **grocery store** 🛒 where the chef gets pre-made ingredients (libraries).
- **Technical detail:**
    - **Node.js**: runtime environment for executing JavaScript outside browsers (built on Chrome's V8 engine).
    - **npm**: package manager for Node, used to install dependencies.

---

6. Git

- **Simple analogy:**
  Git is like the **time machine for your code** ⏳. You can travel back, see who changed what, and work together without overwriting each other's work.
- **Technical detail:**
  Git is a **distributed version control system**.
    - **Distributed**: every developer has the full repo history.
    - Enables branching, merging, collaboration.

---

7. k6

- **Simple analogy:**
  k6 is like a **treadmill stress test** 🏃 for your app. It makes lots of fake users run against your system to see if it can handle the load.
- **Technical detail:**
  k6 is an **open-source load testing tool**.
    - Test scripts written in JavaScript.
    - Useful for generating traffic, measuring latency, throughput, and triggering autoscaling (HPA).

📜 **Key Commands Explained**

1. docker run hello-world

- **Simple:** Runs a “Hello” container to check Docker works.
- **Technical:** Pulls the hello-world image from Docker Hub → starts a container → prints message.
- **Keywords:** *image* (blueprint), *container* (running instance).

---

2. kind create cluster --name precheck

- **Simple:** Builds a practice Kubernetes cluster named precheck.
- **Technical:** Downloads a kindest-node Docker image → creates containers → configures control-plane node.
- **Keywords:** *cluster* (group of nodes), *control-plane* (Kubernetes brain).

---

3. kubectl get nodes

- **Simple:** Shows the “computers” (nodes) in the cluster.
- **Technical:** Asks API Server for Node objects → prints status, roles, version.
- **Keywords:** *node* (worker/control machine), *Ready* (healthy).

---

4. kubectl cluster-info

- **Simple:** Tells you where the cluster brain (API) and services are running.
- **Technical:** Shows API server endpoint + CoreDNS endpoint.

---

5. kubectl create deployment web --image=nginx:alpine

- **Simple:** Deploys a small NGINX webserver. Like asking Kubernetes: “please run this app for me”.
- **Technical:** Creates a **Deployment object** with ReplicaSet managing pods running nginx:alpine.
- **Keywords:** *Deployment* (manages replicas), *Pod* (smallest unit in Kubernetes).

---

6. kubectl expose deployment web --port=80

- **Simple:** Opens the app to others inside the cluster.
- **Technical:** Creates a **Service object** that routes traffic to Pods.
- **Keywords:** *Service* (network endpoint), *port* (entry point).

---

7. kubectl port-forward deploy/web 8080:80

- **Simple:** Like poking a tunnel 🔌 from your laptop → cluster app.
- **Technical:** Forwards traffic from **localhost:8080** to Pod's port 80.

---

8. helm repo add bitnami https://charts.bitnami.com/bitnami

- **Simple:** Adds the Bitnami "app store" to Helm.
- **Technical:** Registers remote repo → Helm can fetch charts from it.

---

9. helm search repo nginx

- **Simple:** Search for nginx apps in the repo.
- **Technical:** Queries Helm index.yaml for matching chart names.

---

10. kind delete cluster --name precheck

- **Simple:** Deletes the practice cluster, cleans up containers.
- **Technical:** Stops and removes all Docker containers & networks related to that cluster.

📜 **Glossary of Technical Terms**

- **Cluster:** Group of machines (nodes) managed as one system.
- **Node:** A single machine (VM/container) inside Kubernetes.
- **Pod:** Smallest deployable unit in Kubernetes (one or more containers).
- **Deployment:** Kubernetes object to manage replicas of Pods.
- **Service:** Network abstraction to expose Pods.
- **API Server:** Central brain that accepts kubectl requests.
- **ReplicaSet:** Ensures desired number of Pods are running.
- **YAML:** Human-readable config format used by Kubernetes.
