☸️ Kubernetes Basics

A High-Level Introduction to Container Orchestration

⏱️ **Duration:** 20 minutes

🤔 What is Kubernetes?

It's a system that **automates** running applications in containers.

- **Manages** deployment, scaling, and failures automatically.
- **Orchestrates** containers across a group of machines.
- Originally from Google, now the industry standard.
- Often called **K8s**.

💡 The Core Philosophy: Cattle, not Pets

A key mindset for understanding Kubernetes.

Pets (Traditional Servers)	Cattle (Modern Containers)
Unique, named servers	Identical, anonymous units
Manually nurtured	Automatically managed
If one fails, you fix it	If one fails, you replace it

🏗️ Core Architecture: The Cluster

A Kubernetes cluster is a set of machines, called **nodes**, that run your applications. It consists of two main parts:

- **Control Plane:** The brain 🧠. It manages the cluster and makes decisions.
- **Nodes:** The workers 💪. They are machines (VMs or physical) that run your application containers.

🏗️ Cluster Diagram

The Control Plane manages the Nodes to run your applications.

```
+---------------------------------+
|         CONTROL PLANE           |
|  (The Brain: API, Scheduler)    |
+---------------------------------+
              |
              | Manages
              |
+-------------+-------------------+
|             |                   |
|   NODE 1    |   NODE 2          |   NODE ...
| (Worker)    | (Worker)          |  (Worker)
| - Pod       | - Pod             |  - Pod
| - Pod       | - Pod             |
```

📦 Pods: The Smallest Unit

A **Pod** is the smallest deployable object in Kubernetes.

- It's a wrapper around one or more containers.
- Containers in a Pod share the same network (IP address) and storage.
- **Pods are ephemeral (disposable).** They can be destroyed and replaced at any time.

📦 Pod Diagram

A Pod can contain one or more containers that work together.

```
+---------------------------+
|          POD              |
|   (Single IP Address)     |
|                           |
| +----------+ +----------+ |
| | Your App | |  Logging | |
| | Container| | Sidecar  | |
| +----------+ +----------+ |
|                           |
+---------------------------+
```

📜 The Declarative Model

In Kubernetes, you don't give commands. You **declare a desired state**.

You tell Kubernetes *WHAT* you want, not *HOW* to do it.

- **You write:** "I want 3 copies of my app running."
- **Kubernetes works to:** Make reality match your declaration.

This is the foundation of "self-healing" infrastructure.

🚀 Managing Pods: Deployments

You rarely create Pods directly. You use a **Deployment** to declare your desired state for them.

- A **Deployment** manages a set of identical Pods.
- **It ensures your desired state is met:**
  - If a Pod crashes, it creates a new one.
  - If a Node fails, it moves the Pods to a healthy Node.
- It handles **rolling updates** and **rollbacks** with zero

🌐 Exposing Pods: Services

A **Service** gives you a stable network endpoint for your unstable Pods.

- **Problem:** Pods are replaced often, so their IP addresses change. How can other apps find them?
- **Solution:** A **Service** provides a single, stable IP address and DNS name. It acts as a load balancer, sending traffic to the healthy Pods.

🌐 Deployment + Service Diagram

A Service directs traffic to the Pods managed by a Deployment.

```
                      +-----------+
                      |           |
Incoming Traffic ---> |  SERVICE  | ---> [Pod 1 (IP: 10.1.1.2)]
                      |(Stable IP)|
                      |           | ---> [Pod 2 (IP: 10.1.1.3)]
                      +-----------+
                        |
                        '---> [Pod 3 (IP: 10.1.1.4)]
```

🤔 Why Use Kubernetes?

Benefit	How Kubernetes Achieves It
Self-Healing	Automatically restarts or replaces failed containers.
Scalability	Scale apps up or down easily, or even automatically.
Portability	Runs the same way on any cloud (AWS, GCP, Azure) or on-premise.
Efficiency	Packs containers smartly to maximize server resource usage

✅ Summary: The Core Flow

1. You have a **Cluster** of **Nodes**.
2. You create a **Deployment** to declare your desired state.
    - *"I want 3 copies of my app container."*
3. The Deployment creates and manages **Pods** for you.
4. You create a **Service** to give your Pods a stable IP address.

This declarative workflow is the foundation of modern, resilient applications.

🙏 Questions?

Next Up: GitOps and Argo CD Introduction