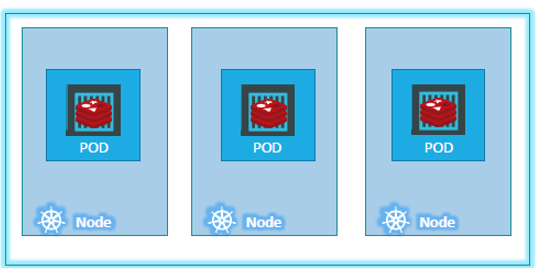
POD



**🔹 What is a Kubernetes Pod?**

A **Pod** is the **smallest deployable unit** in Kubernetes. It represents a **single instance of a running process** in a cluster.

A Pod:

* Wraps one or more **containers**
* Shares the same **network**, **storage**, and **lifecycle**

Think of a Pod as a **wrapper** around one or more containers (usually Docker containers), providing an environment for them to run together.

🔹 Key Characteristics of a Pod

| **Feature** | **Description** |
| --- | --- |
| **Containers** | One or more containers can run in a Pod (most commonly, only one) |
| **Shared Network** | All containers in the same Pod share the **same IP address**, **hostname**, and **network namespace** |
| **Shared Storage** | Volumes can be mounted into containers within the Pod |
| **Tightly Coupled** | Containers in a Pod can communicate via localhost, as they run on the same node |
| **Ephemeral (Immutable)** | Pods are **not durable** by default—if a Pod dies, a new one is created, but it's a different instance |

🔹 Why Multiple Containers in a Pod?

You can run multiple containers in a Pod when they are **tightly coupled** and need to share resources.

This is called a **multi-container Pod**.

**🔹 Pod Lifecycle**

1. **Pending** – The Pod is accepted but not yet scheduled to a node.
2. **Running** – The Pod has been bound to a node and all containers are started.
3. **Succeeded** – All containers terminated successfully.
4. **Failed** – One or more containers failed.
5. **Unknown** – Status couldn't be obtained, usually due to communication error.

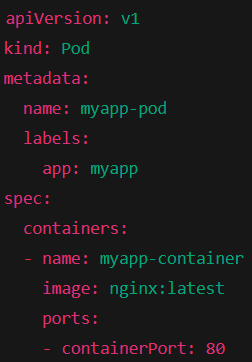
**🔹 How Pods are Created**

Pods are rarely created directly by users. Instead, they are managed by higher-level Kubernetes objects like:

* **Deployments**
* **ReplicaSets**
* **StatefulSets**
* **DaemonSets**

These controllers ensure the desired state of the application is maintained and will recreate Pods as needed.

🔹 Pod YAML Example



You can create it with:



**🔹 Accessing Pods**

* Each Pod gets an **IP address**, but you typically don't access Pods directly.
* Instead, you use **Services** in Kubernetes to expose Pods.
* Services provide stable IPs and DNS names and load-balance traffic to Pods.

**🔹 Volumes in Pods**

Pods can use Kubernetes **Volumes** to persist and share data among containers. Volumes are declared in the Pod spec and mounted in containers.

Example:

A screen shot of a computer

AI-generated content may be incorrect.

**🔹 Limitations of Pods**

* Pods are **ephemeral**; they don't persist after failure (unless managed by a controller).
* Pods cannot span across multiple nodes.
* Scaling Pods manually is not efficient—use Deployments or StatefulSets.

**Types PODs**

**🔶 Single-Container Pods**

A computer screen shot of a computer

AI-generated content may be incorrect.  
 **🔶 Multi-Container Pods**

A **multi-container Pod** is a Pod that contains **more than one container**, which are tightly coupled and need to **cooperate** to function as a single unit.

**🧠 Why use Multi-Container Pods?**

Use multi-container Pods when:

* Containers need to **share a local filesystem** (using volumes).
* They must communicate **via localhost**.
* They are **part of the same logical application**.

A tablet with a diagram of a web hosting

AI-generated content may be incorrect.  
  


**Q.) How application is deployed in the Kubernetes cluster?**

**Example:** Single node Kubernetes cluster with a single POD (where Docker is running)A screenshot of a computer

AI-generated content may be incorrect.

Q.) What if the number of users accessing our application increase?

We need to add additional instances of our application to share the load.

Q.) Do we bring up a new container instance within the same POD?   
  
No! We create a new POD altogether with a new instance of the same application in same Kubernetes system or node.

Q.) What if the user base further increases and our current node has no sufficient capacity?

* Then we can always deploy additional PODs on a new node in the Kubernetes cluster.
* PODs usually have a one-to-one relationship with containers
  + To scale UP we create new PODs and to scale down we delete PODs
  + We do not add additional containers to an existing POD to scale our application.