What is an init container, and how is it different from a normal container?

An **Init Container** in Kubernetes is a specialized container that runs **before** the main application container starts. It is useful for performing setup tasks like:

- Initializing configuration
- Waiting for dependencies (databases, services)
- Setting up secrets or fetching data

Unlike a normal application container, an **Init Container:**
✅ Runs **once** and exits before the main container starts.
✅ Ensures prerequisites are met before the app runs.
✅ Has its own separate image and lifecycle.

Example: Init Container in Action

We will deploy a **Pod** with:
1️⃣ **An Init Container** that waits for 10 seconds (simulating a dependency setup).
2️⃣ **A Main Container** that runs `nginx`.

Step 1: Create a YAML file (`init-container-demo.yaml`)

```
apiVersion: v1
kind: Pod
metadata:
  name: init-container-demo
spec:
  containers:
```

```
  - name: main-app
    image: nginx
    ports:
    - containerPort: 80
  initContainers:
  - name: setup
    image: busybox
    command: ["sh", "-c", "echo 'Initializing...'; sleep 10; echo 'Setup
Complete'"]
```

Step 2: Deploy the Pod

Run this command to apply the configuration:

```
sh
-----------
kubectl apply -f init-container-demo.yaml
```

Command	Purpose
`kubectl get pod init-container-demo`	Check Pod status
`kubectl describe pod init-container-demo`	Get detailed container status
`kubectl get pod init-container-demo -w`	Watch Init Container transition
`kubectl logs init-container-demo -c setup`	View Init Container logss
`kubectl logs init-container-demo -c main-app`	View nginx logs
`kubectl exec -it init-container-demo -c main-app -- sh`	Enter running nginx container

For deleting pod

kubectl delete pod init-container-demo

Step 3: Check Pod Status

```
sh
-----------
kubectl get pods
```

Initially, you will see `Init:0/1`, indicating that the **Init Container is running**.
Once it finishes, the main container (`nginx`) starts.

To check logs:

```
sh
-----------
kubectl logs init-container-demo -c setup
```

You will see:

```
-----------
Initializing...
Setup Complete
```

Summary

- 🔹 The Init Container runs first to simulate a setup.
- 🔹 After it exits successfully, the main `nginx` container starts.
- 🔹 If the Init Container fails, the Pod **won't start** until it succeeds.

This is useful in real-world scenarios like waiting for a database to be ready before starting an application.

Which of the following is a common use case for an Init Container?

A) Serving web traffic to users
B) Running database queries after the application starts
C) Preparing configuration files before the main container starts
D) Handling real-time data processing

✅ **Answer:** C) Preparing configuration files before the main container starts

What happens if an Init Container fails?

A) The Pod continues running with the main container
B) Kubernetes retries the Init Container until it succeeds or the Pod fails
C) The Pod is deleted immediately
D) The Init Container is ignored, and the main container starts

✅ **Answer:** B) Kubernetes retries the Init Container until it succeeds or the Pod fails

How is an Init Container different from a normal application container?

A) It runs continuously along with the main container
B) It has a different runtime than the main container
C) It always restarts after completion
D) It runs first and exits before the main container starts

✅ **Answer:** D) It runs first and exits before the main container starts