What is PreStop Hook ? Why it is important ?

What is a preStop hook in Kubernetes?

The `preStop` hook is a lifecycle event that **executes a command just before the container is terminated** (i.e., before a pod is shut down).
It gives your app time to **finish ongoing work, clean up, or gracefully close connections** before it's killed.

Why is it important?

Imagine your app is handling requests from users. Suddenly, Kubernetes decides to terminate one pod (maybe for an update). If it kills it instantly:

- Some user requests may get dropped mid-way.
- You could face errors or partial processing.

To prevent this, you can use a `preStop` hook to **delay termination and allow cleanup**.

Scenario: E-Commerce App with Cart and Checkout

You have a Node.js app running in Kubernetes. Users are:

- Browsing items to the ca
- Adding items rt
- Clicking **"Checkout"** — the app processes their payment and stores order info in a database.

Now imagine:
Kubernetes decides to terminate a pod running this app — **right when a user is checking out!**

If the pod is killed **instantly**, the order may be:

- Not saved
- Payment in limbo
- User sees an error

That’s a bad customer experience

What can `preStop` do here?

Before shutting down, the app can:

- Finish the current "checkout" request
- Close DB connections gracefully
- Notify the load balancer it's no longer accepting traffic (via readiness probe)

```
yaml

lifecycle:
  preStop:
    exec:
      command: ["/bin/sh", "-c", "sleep 10"]
```

This allows ongoing requests to complete before the pod is terminated.

✅ 1. You delete the Pod (e.g., via `kubectl delete pod mypod`)

- The Pod goes into the **"Terminating"** state.
- Kubernetes **does not immediately kill the container**. It begins a graceful shutdown process.

✅ 2. `preStop` hook is triggered (if defined)

- Kubernetes **runs the** `preStop` **lifecycle hook** defined in the container spec.
- This can be a shell command or an HTTP call.
- Example:

```
yaml

lifecycle:
  preStop:
    exec:
      command: ["/bin/sh", "-c", "echo Shutting down..."]
```

Common Use Cases for preStop Hook

Here are **practical things** that are typically done in a preStop hook:

1. Deregistering from a Load Balancer / Service Mesh

So that the container stops receiving new traffic before it shuts down. curl -X POST http://loadbalancer.local/deregister

2. Sending a shutdown message to an external system

Notify a monitoring service, audit logger, or centralized orchestrator. curl -X POST http://notify-service.local/pod-stopping

3. Flushing logs or metrics
Final logs, trace, or metrics can be sent before the container dies.
Example: Push final Prometheus metrics or log flush to ELK/Datadog.

4. Finishing In-flight Tasks / Cleanup Work
Safely complete any running job or transaction to avoid corruption or partial work. /app/clean-temp-files.sh

5. Unmounting volumes or releasing resources
Useful in stateful apps to ensure data is not left hanging. umount /mnt/data

`preStop` Hook

- **Defined in the Pod spec**, outside of your application code.
- Managed by **Kubernetes**, not your app.
- Runs **just before** the container is terminated.
- Great for handling **external cleanup** like:
  - Deregistering from a **load balancer**
  - Notifying external services
  - Triggering shell scripts or HTTP calls
  - Writing a log or status marker file

✅ 3. Kubernetes sends SIGTERM to the container process

- Right **after** (or simultaneously with) starting the `preStop` hook, Kubernetes **sends a** `SIGTERM` **signal** to the container process (usually PID 1).
- This is a signal to allow the app to shut down gracefully.
- Your app can trap this signal to start **cleanup activities**.

✅ 4. Kubernetes waits for terminationGracePeriodSeconds

- Kubernetes **waits for the** `preStop` **hook and the app to finish gracefully**, for a duration defined in:

```
yaml
CopyEdit
terminationGracePeriodSeconds: 30
```

- During this time:
    - Your `preStop` hook should finish.
    - Your app should ideally exit on its own.
    - If either hangs or takes too long, Kubernetes proceeds to step 5.

✅ 5. Kubernetes sends SIGKILL (if grace period ends)

- If the process **has not exited** by the end of `terminationGracePeriodSeconds`:
    - Kubernetes forcefully kills it with a `SIGKILL`.
    - This immediately stops the container—no more cleanup possible.

✅ 6. Pod is removed from the cluster

- The container is removed.
- The Pod object is deleted from the Kubernetes API.
- Finalizers (if any) are handled.

Python App with Graceful Shutdown (Using Kubernetes preStop Hook)

✅ Step 1: Project Structure

Create a new folder:

```
bash
------------
mkdir kind-prestop-python && cd kind-prestop-python
```

✅ Step 2: Create `app.py`

```
python

# app.py
import signal
import time

def handle_sigterm(signum, frame):
    print("📩 Got termination signal. Starting cleanup...")
    time.sleep(3)
    print("✅ Saved unsaved items to database (simulated)")
    time.sleep(3)
```

```
    print("🛑 Closed DB connection (simulated)")
    print("🚪 Exiting now.")
    exit(0)

signal.signal(signal.SIGTERM, handle_sigterm)

print("✅ Python app started. Waiting...")

# Infinite loop
while True:
    time.sleep(1)
```

✅ Step 3: Create `Dockerfile`

```
Dockerfile
------------
# Dockerfile
FROM python:3.10-slim
WORKDIR /app
COPY app.py .
CMD ["python", "app.py"]
```

✅ Step 4: Create Kind Cluster and Load Docker Image

```
bash
------------
kind create cluster --name prestop-python
docker build -t python-prestop-app:latest .
kind load docker-image python-prestop-app:latest --name prestop-python
```

✅ Step 5: Create Kubernetes Deployment YAML

Create `deployment.yaml`:

```
yaml
------------
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: python-app
  template:
    metadata:
      labels:
        app: python-app
    spec:
      containers:
      - name: python-app
        image: python-prestop-app:latest
        lifecycle:
          preStop:
            exec:
```

```
            command: ["/bin/sh", "-c", "echo 'preStop hook triggered'; sleep
10"]
```

✅ Step 6: Deploy to Kubernetes

```
bash
------------
kubectl apply -f deployment.yaml
```

✅ Step 7: Observe Logs and Test `preStop`

1. Get pod name:

```
bash
------------
kubectl get pods
```

2. Open log stream:

```
bash
------------
kubectl logs -f <pod-name>
```

3. In another terminal, delete the pod:

```
bash
------------
kubectl delete pod <pod-name>
```

You'll see:

```
java
------------
📩 Got termination signal. Starting cleanup...
✅ Saved unsaved items to database (simulated)
🔒 Closed DB connection (simulated)
👋 Exiting now.
```

Kubernetes waits 10 seconds due to `preStop` hook before actually killing the pod, giving our app time to shut down gracefully.

What is the purpose of a PreStop hook in Kubernetes?

A. To run a command after a container starts
B. To delay the start of a container
C. To run a command before a container is terminated
D. To monitor the health of a container

Answer: ✅ C. To run a command before a container is terminated

Which of the following lifecycle hooks does Kubernetes support for containers?

A. postStart
B. preStop
C. onTerminate
D. Both A and B

Answer: ✅ D. Both A and B