***🧑‍🏫 Kubernetes Workload Controllers***

**1. Deployment**

* **What it is:**
  + Manages **stateless applications**.
  + Ensures a specified number of **replicas** (Pods) are always running.
  + Supports **rolling updates** and **rollbacks**.
* **Use case:** Web servers, APIs, frontends.
* **Example (nginx deployment with 3 replicas):**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

namespace: demo

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Check:

kubectl get deployments -n demo

kubectl get pods -n demo

**2. ReplicaSet**

* **What it is:**
  + Ensures a certain number of **identical Pods** are running at any time.
  + Deployments actually **use ReplicaSets under the hood**.
  + Usually, you don’t create ReplicaSets directly (you use Deployments instead).
* **Use case:** If you only need to guarantee a set of Pods without updates/rollbacks.
* **Example (2 replicas of nginx):**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx-rs

namespace: demo

spec:

replicas: 2

selector:

matchLabels:

app: nginx-rs

template:

metadata:

labels:

app: nginx-rs

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Check:

kubectl get rs -n demo

kubectl get pods -n demo

**3. StatefulSet**

* **What it is:**
  + Used for **stateful applications** (databases, Kafka, Redis, etc.).
  + Provides **stable pod identity (DNS names)** and **persistent storage**.
  + Pods are created **in order** and deleted **in reverse order**.
* **Use case:** Databases, queues, apps that need **stable network IDs** and **storage volumes**.
* **Example (3 replicas of nginx with stable identity):**

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

namespace: demo

spec:

serviceName: "nginx-service"

replicas: 3

selector:

matchLabels:

app: nginx-sts

template:

metadata:

labels:

app: nginx-sts

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Check pod names:

kubectl get pods -n demo

You’ll see pods named like:

web-0, web-1, web-2

**4. DaemonSet**

* **What it is:**
  + Ensures **one pod per node** (or specific nodes).
  + Used for **background system services**.
* **Use case:** Logging agents (Fluentd), monitoring agents (Prometheus Node Exporter), networking (CNI plugins).
* **Example (run nginx on every node):**

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: nginx-ds

namespace: demo

spec:

selector:

matchLabels:

app: nginx-ds

template:

metadata:

labels:

app: nginx-ds

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Check:

kubectl get ds -n demo

kubectl get pods -o wide -n demo

👉 You’ll see **one Pod per node**.

**📝 Summary Table**

| **Controller** | **Pods per Node** | **Identity** | **Use Case** | **Scaling** |
| --- | --- | --- | --- | --- |
| **Deployment** | Multiple | Random | Stateless apps (web, APIs) | Yes |
| **ReplicaSet** | Multiple | Random | Low-level control of replicas | Yes |
| **StatefulSet** | Multiple | Stable (pod-0, pod-1…) | Databases, Kafka, Redis | Yes |
| **DaemonSet** | 1 per node | One per node | Monitoring, logging, networking | Auto (per node) |

**Kubernetes Job**

👉 A **Job** runs a Pod **until it completes successfully** (exit code 0).

* Unlike Deployments (which keep Pods alive), Jobs are **meant for tasks that finish**.
* Common use cases:
  + Database migrations
  + Batch processing
  + Sending emails
  + Generating reports

**Important Points:**

1. **restartPolicy must be Never or OnFailure** (not Always).
2. Can define:
   * **completions** → total number of successful Pods required.
   * **parallelism** → how many Pods run at the same time.
   * **backoffLimit** → how many retries before Job fails.
3. Once a Job is complete, the Pod will show Completed.

**✅ Example Job**

apiVersion: batch/v1

kind: Job

metadata:

name: demo-job

namespace: nginx

spec:

completions: 3 # Total successful Pods required

parallelism: 2 # Run 2 Pods at a time

backoffLimit: 4 # Retry max 4 times if Pod fails

template:

spec:

containers:

- name: batch-task

image: busybox:latest

command: ["sh", "-c", "echo Processing data for Himanshu && sleep 5"]

restartPolicy: Never

**How it works:**

* Runs **3 Pods total**. Kubectl apply –f job.yml –n nginx
* At most **2 Pods run in parallel**.
* If a Pod fails, Kubernetes retries up to **4 times**.

**🔹 Kubernetes CronJob**

👉 A **CronJob** is just a **Job on a schedule** (like Linux cron).

* Common use cases:
  + Regular backups
  + Sending daily reports
  + Cleanup scripts
  + Periodic health checks

**Important Points:**

1. schedule field uses **cron syntax** → "\*/5 \* \* \* \*" means every 5 minutes.
2. Each scheduled run **creates a Job**.
3. You can set:
   * **concurrencyPolicy**
     + Allow → allow multiple runs simultaneously (default).
     + Forbid → skip new run if previous is still running.
     + Replace → cancel old run, start new one.
   * **successfulJobsHistoryLimit** → keep history of successful Jobs.
   * **failedJobsHistoryLimit** → keep history of failed Jobs.

**✅ Example CronJob**

apiVersion: batch/v1

kind: CronJob

metadata:

name: backup-cron

namespace: nginx

spec:

schedule: "\*/2 \* \* \* \*" # Run every 2 minutes

concurrencyPolicy: Forbid # Don’t start new if old is running

successfulJobsHistoryLimit: 3 # Keep last 3 successful jobs

failedJobsHistoryLimit: 2 # Keep last 2 failed jobs

jobTemplate:

spec:

template:

spec:

containers:

- name: backup-task

image: busybox:latest

command: ["sh", "-c", "date; echo Running backup job for Himanshu"]

restartPolicy: OnFailure

**How it works:**

* Every **2 minutes**, a Job is created. Kubectl apply –f cron-job.yml –n nginx
* Each Job runs a Pod that prints date + backup message.
* If a Job fails, it retries depending on default backoff.
* Old Job history is cleaned up automatically.

**🔑 Quick Difference: Job vs CronJob**

| **Feature** | **Job** | **CronJob** |
| --- | --- | --- |
| Execution | Runs once (or until completions reached) | Runs on a schedule |
| Use case | Batch tasks (one-time) | Periodic tasks (recurring) |
| Example | Data migration | Daily backup |