KUBERNETES DEPLOYMENT STRATEGIES (WITH REAL-TIME EXAMPLES)

1. Recreate Strategy

Description: Deletes **all old pods first** → then creates new pods.
Causes downtime during the process.

Real-Time Example:
Let's say you are upgrading your backend service (v1 → v2) and a **database schema change is not backward compatible**.
You prefer to stop old pods completely, apply DB migrations, then start new pods.

YAML Example:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-backend
spec:
  strategy:
    type: Recreate
  replicas: 3
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: my-backend:v2
```

Interview Tip:
"I use Recreate when downtime is acceptable — for example, during major DB migrations. This avoids version conflicts."

2. Rolling Update (Default)

Description: Updates pods **gradually** (e.g., one by one) → zero downtime if enough replicas exist. Ensures users always have some pods serving traffic.

Real-Time Example:
You are upgrading your e-commerce frontend from v1 → v2.
Users won't see any downtime — old pods serve requests while new ones are created.

YAML Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-frontend
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  replicas: 4
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: my-frontend:v2
```

Interview Tip:
"Rolling updates are my go-to for production. They give zero downtime upgrades — I just set maxUnavailable and maxSurge to control the rollout speed."

---

3. Blue-Green Deployment

Description:
Two identical environments:

- **Blue** = current version
- **Green** = new version
  When Green is ready, you just **switch service traffic** to it.
  Easy rollback: switch back to Blue.

Real-Time Example: You're deploying a **payment service**.
You deploy v2 as Green, test it with staging traffic, then switch 100% production traffic to it instantly.
If issues appear, you switch back to Blue in seconds.

YAML Example (Service Switch):

```
# Blue Deployment
apiVersion: apps/v1
```

```yaml
kind: Deployment
metadata:
  name: payment-blue
spec:
  replicas: 3
  selector:
    matchLabels:
      app: payment
      version: blue
  template:
    metadata:
      labels:
        app: payment
        version: blue
    spec:
      containers:
      - name: payment
        image: my-payment:v1

# Green Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: payment-green
spec:
  replicas: 3
  selector:
    matchLabels:
      app: payment
      version: green
  template:
    metadata:
      labels:
        app: payment
        version: green
    spec:
      containers:
      - name: payment
        image: my-payment:v2

# Service points to either blue or green
apiVersion: v1
kind: Service
metadata:
  name: payment-service
spec:
  selector:
    app: payment
    version: green  # <-- Switch this to 'blue' if rollback is needed
  ports:
  - port: 80
  targetPort: 8080
```

Interview Tip:
"Blue-Green gives me a quick rollback option. I just change the Service selector to switch between versions."

---

4. Canary Deployment

Description: Releases new version to a **small subset of users first** (say 10%), then gradually increase traffic if no issues are found.

Real-Time Example: You deploy a new recommendation engine. Start with 10% traffic to test errors and performance. If everything is stable, scale it to 50% → then 100%.

YAML Example (Partial Rollout):

```
# Old Version
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-stable
spec:
  replicas: 8
  selector:
    matchLabels:
      app: my-app
      version: stable
  template:
    metadata:
      labels:
        app: my-app
        version: stable
    spec:
      containers:
      - name: my-app
        image: my-app:v1

# Canary Version
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-canary
spec:
  replicas: 2  # Only 20% of traffic
  selector:
    matchLabels:
      app: my-app
      version: canary
  template:
```

```yaml
  metadata:
    labels:
      app: my-app
      version: canary
  spec:
    containers:
    - name: my-app
      image: my-app:v2

# Service routes to both (using labels)
apiVersion: v1
kind: Service
metadata:
  name: my-app-service
spec:
  selector:
    app: my-app
  ports:
  - port: 80
```

Interview Tip:
"I use Canary for safe rollouts. For example, I deploy v2 to 10% of pods first, monitor metrics, then gradually increase."

---

5. Shadow Deployment

Description: New version gets a **copy of real production traffic**, but responses are **not returned to users**.
Used for performance or load testing.

Real-Time Example:
Testing a new ML model with production traffic before officially releasing it.

How It Works:

- Old service handles real users.
- A proxy (like Istio, Nginx, or Envoy) duplicates requests and sends to the new version.
- New version logs responses but does not impact users.

Interview Tip:
"I use Shadow when I want to test under real-world traffic without impacting users — e.g., testing a new ML model latency."