You're tasked with deploying an application that needs zero downtime during upgrades. How would you design the Kubernetes deployment strategy for this?

✅ 1. Use Rolling Updates (Default Strategy)

Kubernetes Deployments support rolling updates out of the box, ensuring that the application is upgraded gradually without downtime.

Configuration:

```
yaml
------------
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 1
    maxUnavailable: 0
```

- `maxSurge: 1` allows creating one extra pod beyond the desired count during updates
  - `maxUnavailable: 0` ensures zero downtime by requiring that a replacement pod is available before removing an old one

---

✅ 2. Readiness Probes

Ensure you configure **readiness probes** properly so that a new pod is only added to the load balancer after it is fully ready.

```
yaml
```

```
-----------
readinessProbe:
  httpGet:
    path: /healthz
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 10
```

This prevents traffic from being routed to a pod that is still starting up.

✅ 3. PreStop Hook

Use `preStop` hooks to gracefully shut down your application and drain in-flight requests.

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.

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

✅ 4. PodDisruptionBudgets (PDBs)

Set up a **PDB** to control the number of pods that can be voluntarily disrupted, ensuring availability during voluntary disruptions like node maintenance.

```
yaml
-----------
minAvailable: 90%
```

✅ 5. Blue-Green or Canary Deployment (Advanced Zero-Downtime Strategies)

🔵 Blue-Green Deployment:

- Deploy the new version as a separate environment.
- After verification, switch the service from old to new.

This gives **instant rollback** if needed.

🟡 Canary Deployment:

- Gradually shift a percentage of traffic to the new version.
- Monitor performance and errors.
- If all is good, shift 100% traffic.

You can implement this using:

- Service mesh (like Istio)

- Tools like **Argo Rollouts**, **Flagger**

✅ 6. Automation & Observability

- Monitor with Prometheus, Grafana, or ELK stack.
- Automate health checks and rollbacks on failure.
- Integrate CI/CD pipelines for controlled rollout.

You're tasked with deploying an application that requires zero downtime during upgrades. Which of the following Kubernetes strategies best ensures this goal?

A. Use a Recreate deployment strategy with no readiness or liveness probes
B. Use a RollingUpdate strategy with properly configured readiness probes and graceful shutdown via PreStop hooks and SIGTERM handling
C. Use a Blue-Green deployment but scale down the old version before scaling up the new one
D. Use a Job to manage the application lifecycle manually