What is a Pod Disruption Budget (PDB)? Why Do We Need a Pod Disruption Budget (PDB) If Deployments and Replicas Ensure Availability?

A **Pod Disruption Budget (PDB)** in Kubernetes is a policy that ensures a minimum number of Pods remain available during **voluntary disruptions**, such as:

- **Node maintenance** (e.g., upgrades, scaling down nodes)
- **Pod evictions** (due to cluster autoscaling or administrator actions)

How Does PDB Work?

PDB does **not** prevent all evictions but ensures that at least a specified number of Pods remain running.

You can define this using either:

- `minAvailable` → Minimum number of Pods that must remain running.
- `maxUnavailable` → Maximum number of Pods that can be unavailable at a time.

Example PDB Definition

```
yaml
------
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: my-app-pdb
spec:
  minAvailable: 1  # Ensures at least 1 Pod is always running
  selector:
    matchLabels:
      app: my-app
```

OR

```
yaml
------
spec:
  maxUnavailable: 1  # Allows 1 Pod to be evicted at a time
```

When to Use PDB?

✅ For **high-availability applications**, ensuring **at least one replica** is always available.

✅ For **database clusters (e.g., MongoDB, PostgreSQL)**, where disruptions must be controlled.

✅ To **prevent downtime during Kubernetes upgrades**.

Limitations of PDB

❌ **Does not prevent involuntary disruptions** (e.g., node failures).

❌ **Works only with managed controllers** (e.g., Deployments, StatefulSets, but not standalone Pods).

Example

🔹 Step 1: Create a Deployment with Multiple Replicas

We'll create a **Deployment** with 3 replicas of an Nginx Pod.

```
yaml
------
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3  # Running 3 Pods
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
```

Apply the deployment:

```
sh
------
kubectl apply -f nginx-deployment.yaml
```

🔹 Step 2: Create a Pod Disruption Budget (PDB)

This PDB ensures **at least 2 Pods remain running** during voluntary disruptions.

```
yaml
------
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: nginx-pdb
spec:
  minAvailable: 2  # At least 2 Pods must be running
  selector:
    matchLabels:
      app: nginx
```

Apply the PDB:

```
sh
------
kubectl apply -f nginx-pdb.yaml
```

🔷 Step 3: Check Pod and PDB Status

Verify that the Deployment and PDB are created:

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

You should see something like:

```
css
------
NAME         MIN AVAILABLE   MAX UNAVAILABLE   ALLOWED DISRUPTIONS   AGE
nginx-pdb    2              N/A               1                     10s
```

- `ALLOWED DISRUPTIONS: 1` → Only **1 Pod** can be disrupted at a time.

🔷 Step 4: Try to Evict a Pod

Attempt to **delete a Pod manually**:

```
sh
------
kubectl delete pod <nginx-pod-name>
```

- If **at least 2 Pods remain**, Kubernetes allows the deletion.
- If deleting a Pod would bring the running Pods below **2**, **Kubernetes blocks the eviction**.

To simulate **a node drain** (which tries to evict all Pods on a node):

```
sh
------
kubectl drain <node-name> --ignore-daemonsets
```

- **If the PDB condition is violated**, the drain will be **blocked**.

🔷 Step 5: Clean Up (Optional)

If you want to remove everything:

```
sh
------
kubectl delete deployment nginx-deployment
kubectl delete pdb nginx-pdb
```

🤔 Why Do We Need a Pod Disruption Budget (PDB) If Deployments and Replicas Ensure Availability?

You're absolutely right that **Deployments and ReplicaSets** automatically recreate Pods if they are deleted. However, **PDB serves a different purpose**—it **controls how many Pods can be disrupted at the same time during voluntary disruptions** like:

- **Cluster maintenance (Node drain, OS updates, scaling down)**
- **Kubernetes cluster upgrades**
- **Pod eviction during auto-scaling**
- **Node preemption in cloud environments (e.g., spot instances)**

✅ What You Learned?

- **PDB prevents Kubernetes from removing too many Pods at once.**
- **PDB only applies to voluntary disruptions (e.g., `kubectl drain`).**
- **Involuntary disruptions (e.g., node failure) are NOT prevented by PDB.**

1. What is the main purpose of a Pod Disruption Budget (PDB)?

a) To prevent Pods from restarting
b) To ensure a minimum number of Pods remain available during voluntary disruptions
c) To limit the number of Pods that can run in a namespace
d) To prevent manual Pod deletion

✅ **Answer: b**

---

2. Which of the following is considered a *voluntary disruption* that PDB protects against?

a) Node crashes
b) Kubernetes cluster upgrades
c) Out-of-memory (OOM) kills
d) Hardware failure

✅ **Answer: b**