**🔧 Pod Disruption Budget (PDB) in Kubernetes**

**📌 What is a Pod Disruption Budget?**

A **Pod Disruption Budget (PDB)** is a **Kubernetes policy** that ensures a **minimum number of pods remain available** during **voluntary disruptions**.

💡 It does **not prevent** pod failures. It only controls **how many can be disrupted simultaneously** for **voluntary disruptions** like upgrades, maintenance, or draining nodes.

**🌱 Why PDB is Useful (Real-World Use Cases)**

**✅ Use Cases:**

1. **High Availability (HA)**:
   * You have a replicated application (e.g., replicas: 3) and want to ensure **at least 2 pods stay available** during node upgrades or auto-scaling.
2. **Cluster Upgrades**:
   * When using tools like kubeadm, GKE, EKS, or AKS, PDBs are respected during **automated upgrades** or **node rotations**.
3. **Pod Evictions during Node Drain**:
   * kubectl drain honors PDBs — it will **block eviction** of pods if evicting them would violate the PDB.
4. **Pod Autoscaling or Rolling Updates**:
   * Prevents too many replicas from being taken offline at once during a **Deployment rollout**.

**🛠️ How PDB Works Internally**

**🔄 Voluntary vs Involuntary Disruptions**

| **Type** | **Examples** | **PDB Applies?** |
| --- | --- | --- |
| Voluntary | Node drain, rolling updates, scaling, upgrade | ✅ Yes |
| Involuntary | Node crash, hardware failure, OOMKill | ❌ No |

**💡 Mechanism:**

* When a **voluntary disruption** (like node drain) happens, the **Eviction API** is used to request pod termination.
* The **Eviction API** checks whether evicting the pod would violate **any matching PDB**.
* If it would, the **eviction is denied**.

**📄 PDB Specification Syntax**

apiVersion: policy/v1

kind: PodDisruptionBudget

metadata:

name: my-app-pdb

spec:

minAvailable: 2 # OR use maxUnavailable

selector:

matchLabels:

app: my-app

**🧮 Key Fields in PDB**

| **Field** | **Description** |
| --- | --- |
| minAvailable | Minimum number of pods that must be available at all times |
| maxUnavailable | Maximum number of pods that can be unavailable at a time |
| selector | Label selector to target the group of pods this PDB applies to |

⚠️ You can define **only one** of minAvailable or maxUnavailable.

**🧪 Example Scenario**

Let’s say you have a Deployment with:

* 5 replicas
* A PDB with minAvailable: 4

If someone drains a node with 2 of those pods:

* Only 1 pod will be evicted at a time.
* The second eviction is **blocked** until at least 4 pods are **running and ready**.

**🧩 PDB and Eviction API Workflow**

1. You run kubectl drain node-x
2. Kubernetes tries to **evict** the pods on that node.
3. For each pod:
   * Kubernetes checks **matching PDBs**
   * Checks if evicting the pod **violates the PDB budget**
   * If yes → **Eviction is blocked**
   * If no → Pod is evicted

**🔍 How to Monitor PDBs**

**✅ View PDBs:**

kubectl get pdb

**📝 Output Example:**

NAME MIN AVAILABLE ALLOWED DISRUPTIONS AGE

my-app-pdb 4 1 10m

**✅ Detailed Status:**

kubectl describe pdb my-app-pdb

This shows:

* Total matching pods
* Current healthy (available) pods
* Number of **allowed disruptions**
* Which disruptions are **blocked**

**📊 minAvailable vs maxUnavailable**

| **Attribute** | **Meaning** |
| --- | --- |
| minAvailable: 3 | At least 3 pods **must be available** at all times |
| maxUnavailable: 1 | At most 1 pod **can be unavailable** at a time |

💡 Use maxUnavailable when using autoscaling or varying replica counts.  
💡 Use minAvailable when you know your exact availability requirements.

**🧠 Advanced Concepts**

**1. Percentage Values**

You can use percentages (in string format):

minAvailable: "50%"

# or

maxUnavailable: "25%"

This dynamically calculates the allowed/unavailable count based on current replica count.

**2. Multiple PDBs**

* If **multiple PDBs** match the same pod (via label selectors), **all of them must allow the disruption**.
* Be cautious — overlapping PDBs can lead to stricter-than-expected blocking behavior.

**3. PDB with StatefulSets**

PDBs work with StatefulSets as well. However:

* StatefulSets already have sequential pod termination behavior.
* Combining PDB with StatefulSet can cause **slow rolling updates** if minAvailable is too high.

**4. PDB Limitations**

| **Limitation** | **Explanation** |
| --- | --- |
| Doesn’t prevent evictions | Only **blocks voluntary evictions** that would violate budget. |
| No enforcement on single replicas | minAvailable: 1 on a pod with 1 replica means **no disruption allowed ever** |
| Not effective for involuntary failures | Node crash or OOMKill still kills the pod regardless of PDB |

**⚠️ Common Mistakes**

| **Mistake** | **Why It's a Problem** |
| --- | --- |
| Using PDB on single-replica workloads | Blocks all voluntary disruptions |
| Forgetting to match labels properly | PDB doesn't apply to any pods |
| Misconfiguring percentage formats | "50" instead of "50%" → causes errors |
| Using both minAvailable and maxUnavailable | Not allowed; use only one |

**✅ Best Practices**

* Use PDBs on **high-availability workloads (replicated pods)** only.
* Always **test disruption behavior** before production upgrades.
* Use **maxUnavailable with autoscaling**, as it adapts to replica changes.
* **Monitor allowedDisruptions** before attempting maintenance.
* Ensure **label selectors** in PDB match your pods exactly.

**📌 Summary**

| **Topic** | **Description** |
| --- | --- |
| Purpose | Prevents too many voluntary pod disruptions |
| Controlled via | minAvailable or maxUnavailable |
| Applies to | Voluntary disruptions only (node drain, upgrade) |
| Works with | Deployments, StatefulSets, ReplicaSets |
| Enforced via | Eviction API |
| Monitored by | kubectl get/describe pdb |