# Interview questions - Disaster Recovery in Practice

## 1. What’s the difference between failover and backup?

- **Backup** is the process of copying and storing data to protect against loss or corruption. It helps you **restore** data to a previous state but doesn’t ensure system availability during an outage.

- **Failover** refers to **automatic switching** to a redundant or standby system when a primary system fails. It keeps your services **running** with minimal downtime.

- **Key difference**: Backups are for **data recovery**, failover is for **service continuity**.

- **Best practice**: Use **both** — backups for restoring lost data, failover for keeping systems live.

---

## 2. How do you design DR for a high-traffic web app?

- **Start with business requirements**: Define RTO (Recovery Time Objective) and RPO (Recovery Point Objective).

- **Use multi-region deployment**:

    - Deploy application servers and databases in **active-active** or **active-passive** mode.

    - Replicate data asynchronously/synchronously based on consistency needs.

- **Implement failover**:

    - DNS-based failover or load balancer-based routing (e.g., AWS Route 53, GCP Global LB).

    - Use health checks to detect failure and trigger failover automatically.

- **Add data backups**: Frequent snapshots, versioning, and offsite storage.

- **Automate** recovery and perform **DR drills** to validate response plans.

## 3. What is RTO/RPO, and how do you optimize them?

- **RTO (Recovery Time Objective)**: Maximum acceptable time to **restore service** after a failure.
- **RPO (Recovery Point Objective)**: Maximum acceptable amount of **data loss** measured in time (e.g., 5 minutes of data).
- **To optimize**:
  - **Lower RTO** with automated failover, container orchestration, or hot standby environments.
  - **Lower RPO** with real-time or near real-time data replication (e.g., log shipping, CDC).
  - Balance against **cost and complexity** — shorter RTO/RPO means higher infra and operational cost.

---

## 4. What are challenges with geo-distributed DR systems?

- **Data consistency**: Keeping data in sync across regions is difficult, especially under network partition.
- **Latency**: Data replication across regions introduces delays.
- **Split-brain scenarios**: Without proper coordination, different regions might act as "primary," causing inconsistency.
- **Data locality & compliance**: Regulations (e.g., GDPR) may restrict storing user data in certain regions.
- **Failover orchestration**: Switching traffic, promoting standby DBs, and syncing state can be complex.

---

## 5. Explain quorum-based design in distributed recovery.

- **Quorum-based design** ensures decisions (e.g., leader election, write confirmation) are made by **majority consensus** of nodes.

- Helps prevent **inconsistent data** or **split-brain** during failover.
- Example:
  - In a 5-node cluster, at least 3 nodes must agree (quorum) before proceeding with writes or electing a new leader.
- **Used in**:
  - Consensus protocols like **Paxos**, **Raft**
  - Systems like **etcd**, **Zookeeper**, **CockroachDB**
- Ensures **safe recovery** and **high availability** even during regional failures.