Interview Questions - Scaling strategies

📘 Interview Question 1: What is the difference between horizontal and vertical scaling?

Answer:

- **Horizontal Scaling** (Scaling Out): Adding more machines or instances to distribute the load.
  Example: Adding more application servers behind a load balancer.
  ✅ Scales well for web apps and microservices.
  ⚠️ Adds complexity in state management, coordination, and deployment.

- **Vertical Scaling** (Scaling Up): Increasing the resources (CPU, RAM, disk) of a single server.
  Example: Upgrading a database server from 16GB RAM to 64GB RAM.
  ✅ Simpler and faster to implement.
  ⚠️ Has physical/OS limits and can be a single point of failure.

📘 Interview Question 2: What is diagonal scaling and when is it a good idea?

Answer:

Diagonal Scaling is a hybrid approach:

- Start with **vertical scaling** (easier, cheaper for small scale).

- Then move to **horizontal scaling** as load increases beyond a single machine's limits.

✅ **Use cases:**

- Startups: Begin with minimal infra and gradually scale horizontally as demand grows.

- Systems with burst traffic: Use vertical scaling for fast reactions, and horizontal for long-term elasticity.

This approach balances **simplicity + future-proofing**, avoiding premature complexity.

📘 Interview Question 3: What are the trade-offs between horizontal and vertical scaling in terms of performance and complexity?

Answer:

Factor	Horizontal Scaling	Vertical Scaling
Performance	Can scale almost linearly (with effort)	Limited by single machine capacity
Complexity	Requires distributed system design	Easier to manage initially
Fault Tolerance	More resilient (failover possible)	Single point of failure
Cost	Higher operational cost & complexity	High-capacity machines are expensive
Deployment	Requires orchestration & load balancing	Simple deploys, fewer moving parts

Summary:

- Horizontal = More scalable, resilient, but complex.
- Vertical = Quick wins, limited long-term.

📘 Interview Question 4: Can you describe a scenario where horizontal scaling wouldn't help?

Answer:

Horizontal scaling won't help when:

- The workload **isn't parallelizable**. Example: A legacy monolithic app that relies on shared state or global locks.
- There's a **non-distributable bottleneck**, like a single-threaded operation or a relational DB that doesn't scale well horizontally.

- There's a **stateful service** without session persistence or sticky sessions in the load balancer.

Real-world example:

- Scaling a PostgreSQL database for analytics queries — simply adding more nodes won't help unless sharding or read replicas are implemented.

📘 Interview Question 5: When would you choose vertical scaling over horizontal?

Answer:

Choose **vertical scaling** when:

- You need a **quick fix** without refactoring code.
- The system is **monolithic** or tightly coupled.
- Your **team lacks experience** with distributed systems.
- You're in the **early stages** of a project/startup.
- The system has **low concurrency needs** and doesn't warrant added complexity.

✅ Vertical scaling is ideal when you **don't need high elasticity** or can't justify the cost/complexity of a distributed architecture.

📘 Interview Question 6: What challenges arise in horizontal scaling and how would you solve them?

Answer:

Common Challenges:

1. **State Management**
    - Solution: Use external stores like Redis or Memcached for sessions.

2. **Data Consistency**
    - Solution: Implement distributed transactions, eventual consistency patterns, or use CQRS.
3. **Load Distribution**
    - Solution: Use smart load balancers (e.g., with least-connections or IP hashing).
4. **Service Discovery**
    - Solution: Use tools like Consul, Eureka, or cloud-native DNS-based discovery.
5. **Deployment & Orchestration**
    - Solution: Use containers (Docker) and orchestrators like Kubernetes for managing instances.
6. **Increased Latency**
    - Solution: Minimize inter-service calls, implement caching and optimize communication paths.