Interview Questions - Scalability

📘 Interview Question 1: What does scalability mean in the context of system design?

Answer:
Scalability is the ability of a system to handle increased load—whether that's more users, more data, or more traffic—without compromising performance or reliability. In system design, this often means the system can grow linearly or elastically with demand, using techniques like horizontal scaling (adding more machines) or vertical scaling (upgrading hardware). A scalable system maintains responsiveness, throughput, and availability as demand increases.

📘 Interview Question 2: Can you explain a real-world example where scalability was critical to success or failure?

Answer:
One famous example is **Twitter**. Initially, Twitter struggled to scale and became known for its "fail whale" downtime during traffic surges. Their early monolithic architecture couldn't handle rapid user growth. They later migrated to a distributed, microservices-based architecture to enable horizontal scaling. This transition helped Twitter become more stable and responsive at scale.

Alternatively, **Zoom** successfully scaled during the COVID-19 pandemic, going from 10 million to over 300 million daily participants. Their use of cloud-native infrastructure and autoscaling allowed them to absorb the sudden spike in demand.

📘 Interview Question 3: What are the main challenges systems face as they scale?

Answer: Some of the biggest challenges include:

- **Latency:** More components and network hops introduce delays.
- **Bottlenecks:** A single overloaded component can affect the entire system.

- **Downtime:** More nodes and dependencies increase the risk of failure.
- **Cost:** Scaling up infrastructure, especially in the cloud, can become expensive. These challenges require proactive architectural planning, monitoring, and scaling policies to handle gracefully.

📘 Interview Question 4: How would you identify a bottleneck in a scalable architecture?

Answer:
To identify bottlenecks:

1. Use **observability tools** (like Prometheus, Grafana, Datadog) to monitor performance metrics.
2. Look for **sudden spikes** in CPU, memory, or latency.
3. Trace requests end-to-end using **distributed tracing** tools like OpenTelemetry or Jaeger.
4. Use **load testing tools** (like JMeter or k6) to simulate traffic and see where degradation begins.
5. Identify services or layers where **queue lengths grow** or **response times increase** disproportionately.

📘 Interview Question 5: Why does latency increase with scale, and how can you mitigate it?

Answer:
As systems scale:

- More services are added → more network calls
- Data is sharded or partitioned → aggregation needed
- Dependency chains grow → longer request paths

Mitigation strategies:

- **Caching** frequently accessed data (e.g., Redis)
- **Asynchronous processing** for non-critical tasks (queues, background jobs)
- **Reducing network hops** via edge computing or content delivery networks (CDNs)
- **Optimizing DB queries** and reducing cross-service chatter

---

📘 Interview Question 6: How do you balance scalability with cost in cloud-based systems?

Answer:
Balancing scalability with cost involves:

- Using **autoscaling with upper/lower bounds** to prevent overprovisioning.
- Choosing **serverless or FaaS** (like AWS Lambda) for event-driven workloads.
- Implementing **tiered caching** to reduce load on primary databases.
- Leveraging **spot or reserved instances** where possible.
- Monitoring and **right-sizing infrastructure regularly** based on usage patterns. A cost-effective system scales efficiently while only using resources when needed.