Interview questions - Autoscaling & Cloud Best Practices

Conceptual Understanding

Q1. What is autoscaling, and why is it important in distributed systems?
A: Autoscaling is the automatic adjustment of compute resources based on current demand. It's critical in distributed systems because it ensures:

- High availability by scaling out during peak traffic.
- Cost-efficiency by scaling in during idle times.
- Better user experience by minimizing latency and avoiding overload.

Q2. What's the difference between horizontal and vertical scaling?
A:

- **Horizontal Scaling (Scale Out/In):** Adding or removing instances (e.g., more servers, pods).
  ✅ More resilient, better for stateless services.
- **Vertical Scaling (Scale Up/Down):** Increasing or decreasing the resources (CPU/RAM) of a single machine.
  ⚠️ Limited by hardware and often requires downtime.

Q3. How does predictive autoscaling work?
A: Predictive autoscaling uses historical usage patterns and machine learning to forecast future demand. Based on this prediction:

- Resources are provisioned in advance.
- Example: AWS Predictive Scaling uses past traffic patterns to scale EC2 instances proactively before load increases.

⚙️ Cloud-Specific Scenarios

Q4. How does autoscaling work in AWS/Azure/GCP?
A:

- **AWS:** Uses Auto Scaling Groups (ASGs) tied to CloudWatch metrics. Works with EC2, ECS, EKS, Lambda.
- **Azure:** Uses Virtual Machine Scale Sets (VMSS), App Service Plans, and AKS with rules from Azure Monitor.
- **GCP:** Uses Managed Instance Groups (MIGs), Cloud Functions, and Cloud Run with GCP Monitoring. GKE uses Horizontal Pod Autoscaler (HPA).

Q5. How would you set up autoscaling for a containerized application?
A:

- Use an orchestrator like **Kubernetes** or **ECS/EKS/AKS**.
- Set up **Horizontal Pod Autoscaler** based on CPU usage or custom metrics.
- Use a **load balancer** to distribute traffic across scaled pods.
- Configure monitoring and alerting to validate scaling behavior.

Q6. What metrics would you monitor for effective autoscaling?
A:

- CPU and memory usage
- Request rate per instance
- Queue depth (e.g., SQS, Kafka)
- Latency and error rates
- Custom business metrics (e.g., active users, transactions/sec)

💸 Best Practices

Q7. How can you ensure cost optimization when implementing autoscaling?
A:

- Set scaling thresholds wisely to avoid unnecessary scaling.
- Use **spot/preemptible instances** for non-critical workloads.
- Employ **scale-to-zero** for idle services (Cloud Run, Lambda).
- Regularly **rightsize** instances based on usage.
- Use **budgets and alerts** to monitor cloud spending.

Q8. What are some challenges with autoscaling in real-time systems?
A:

- **Scaling latency**: Time taken to spin up new instances may cause temporary lag.
- **Cold starts**: Especially in serverless platforms like Lambda or Cloud Functions.
- **Over-provisioning**: Can lead to higher costs.
- **Under-provisioning**: If thresholds are too high or scaling is delayed, it can cause outages.