AWS Senior Developer Specialist Solutions Architect (DevTx) - Interview Questions & Answers

Overview

Experience: 15 years

Expertise: C++, Go, System Design, Kubernetes, Microservices, IBM Cloud

---

1. How do you approach designing a scalable microservices architecture?
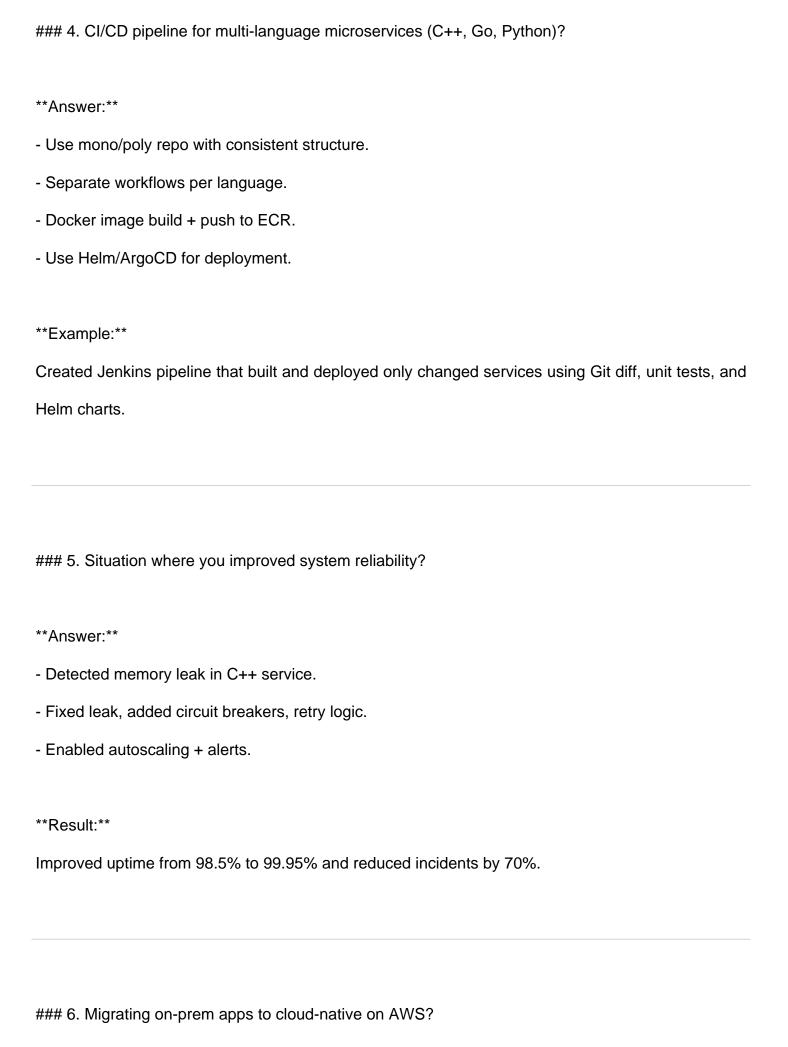
Answer:

- Understand business requirements and SLAs.

- Decompose services by business capability.

- Use REST/gRPC APIs with API Gateway.

- Implement service discovery (Consul or K8s DNS).

- Ensure each service has its own data (eventual consistency).

- Use resilience patterns: circuit breakers, retries, timeouts.

- Add observability: logging, metrics (Prometheus), tracing (Jaeger).
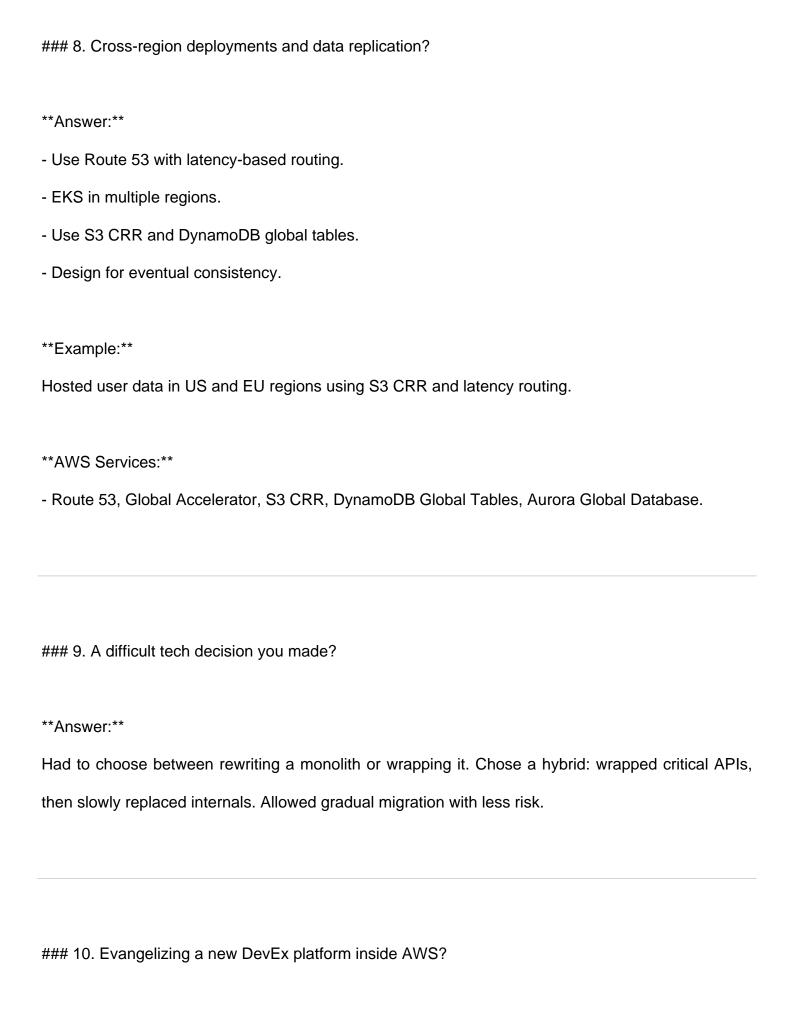
Example:

At IBM Cloud, I designed a microservices-based observability platform with decoupled ingestion, processing, and visualization services using Kafka and Kubernetes.

---

2. How would you containerize and deploy a C++ application on Kubernetes?

Answer:

1. Use a multi-stage Dockerfile to reduce image size.

2. Write K8s manifests or Helm charts.

3. Automate CI/CD with Jenkins/GitHub Actions.

4. Monitor using Prometheus and set liveness probes.

Example:

Containerized a C++ telco backend with Alpine base image. Deployed on EKS with autoscaling based on CPU/memory.

---

3. How do you handle stateful workloads in Kubernetes?

Answer:

- Use StatefulSets for persistent identity.

- Attach PVCs backed by AWS EBS.

- Configure headless services.

- Ensure readiness/liveness probes are accurate.

Example:

Deployed Cassandra via StatefulSet in OpenShift with EBS-backed volumes and proper quorum management.

---

4. CI/CD pipeline for multi-language microservices (C++, Go, Python)?

Answer:

- Use mono/poly repo with consistent structure.

- Separate workflows per language.

- Docker image build + push to ECR.

- Use Helm/ArgoCD for deployment.

Example:

Created Jenkins pipeline that built and deployed only changed services using Git diff, unit tests, and

Helm charts.

---

5. Situation where you improved system reliability?

Answer:

- Detected memory leak in C++ service.

- Fixed leak, added circuit breakers, retry logic.

- Enabled autoscaling + alerts.

Result:

Improved uptime from 98.5% to 99.95% and reduced incidents by 70%.

---

6. Migrating on-prem apps to cloud-native on AWS?

Answer:

- Assess architecture, containerize legacy apps.

- Refactor to microservices where feasible.

- Deploy on EKS or ECS.

- Use AWS-native services: S3, DynamoDB, SQS.

- Implement CI/CD, monitoring (CloudWatch, X-Ray), IAM policies.

Example:

Migrated Go-based logging system from on-prem to AWS. Used EKS, S3 for logs, and reduced cost by 40%.

---

7. Designing secure cloud-native applications?

Answer:

- Enforce least privilege IAM roles.

- Manage secrets via Secrets Manager/Vault.

- Use private subnets, SGs, NACLs.

- TLS for all communications.

- Use Trivy for container image scanning.

AWS Services:

- IAM, Secrets Manager, AWS WAF, Shield, KMS for encryption, GuardDuty for threat detection.

---

8. Cross-region deployments and data replication?

Answer:

- Use Route 53 with latency-based routing.

- EKS in multiple regions.

- Use S3 CRR and DynamoDB global tables.

- Design for eventual consistency.

Example:

Hosted user data in US and EU regions using S3 CRR and latency routing.

AWS Services:

- Route 53, Global Accelerator, S3 CRR, DynamoDB Global Tables, Aurora Global Database.

---

9. A difficult tech decision you made?

Answer:

Had to choose between rewriting a monolith or wrapping it. Chose a hybrid: wrapped critical APIs, then slowly replaced internals. Allowed gradual migration with less risk.

---

10. Evangelizing a new DevEx platform inside AWS?

Answer:

- Identify developer pain points.

- Build a demo or POC to show benefits.

- Involve internal champions.

- Provide tooling: SDKs, CLI, IDE plugins.

- Track DORA metrics and satisfaction.

AWS Tools:

- AWS Cloud9, CDK, SAM, Developer Tools suite (CodeCommit, CodePipeline, CodeBuild, CodeDeploy), AWS SDKs, CloudWatch RUM.