Scenario 1: Your Spring Boot app is slow during peak traffic. How do you fix it on AWS?

Answer: Use ALB + Auto Scaling, enable CloudWatch metrics, add Redis caching, and optimize DB queries.

Scenario 2: How do you achieve zero-downtime deployment for Java microservices?

Answer: Use Blue-Green or Canary deployment with CodeDeploy, ECS rolling updates, or ALB target groups.

Scenario 3: How do you securely upload files from Java to AWS?

Answer: Use Amazon S3 with IAM roles, pre-signed URLs, and server-side encryption.

Scenario 4: Your RDS database goes down. What is your recovery strategy?

Answer: Enable Multi-AZ, automated backups, and snapshots.

Scenario 5: High DB read load is impacting performance. What do you do?

Answer: Add read replicas and caching using ElastiCache.

Scenario 6: How do you store DB credentials securely?

Answer: Use AWS Secrets Manager or SSM Parameter Store with IAM roles.

Scenario 7: Microservices must communicate asynchronously. Solution?

Answer: Use SQS queues, SNS topics, or EventBridge.

Scenario 8: One service failure should not affect others. How?

Answer: Use circuit breakers, retries, and asynchronous messaging.

Scenario 9: How do you monitor Java applications in AWS?

Answer: CloudWatch logs, metrics, alarms, and distributed tracing.

Scenario 10: EC2 instance terminated accidentally. How to avoid downtime?

Answer: Use Auto Scaling Group with desired capacity > 1.

Scenario 11: How do you deploy Dockerized Spring Boot apps?

Answer: Use ECS Fargate or EKS with ECR images.

Scenario 12: How do you restrict S3 access to backend only?

Answer: Use IAM roles and bucket policies.

Scenario 13: How do you protect APIs from abuse?

Answer: Use API Gateway throttling and AWS WAF.

Scenario 14: How do you version APIs?

Answer: Use URL-based or header-based versioning.

Scenario 15: Need to process millions of messages. Architecture?

Answer: Event-driven using SQS, SNS, or Kafka (MSK).

Scenario 16: How do you build CI/CD for Java apps?

Answer: Use CodePipeline, CodeBuild, CodeDeploy.

Scenario 17: Lambda timeout issue. What do you do?

Answer: Optimize code, increase timeout, or move to ECS.

Scenario 18: How do you ensure high availability?

Answer: Multi-AZ deployment with ALB and Auto Scaling.

Scenario 19: How do you reduce AWS cost?

Answer: Use Spot/Reserved instances and auto-scaling.

Scenario 20: Disaster recovery approach?

Answer: Backups, warm standby, and cross-region replication.

Scenario 21: Java service needs caching. What AWS service?

Answer: Use ElastiCache with Redis.

Scenario 22: Spring Boot app config differs per environment. How handle?

Answer: Use Parameter Store and Spring profiles.

Scenario 23: Need temporary AWS access for Java app?

Answer: Use IAM roles with STS tokens.

Scenario 24: How do you encrypt sensitive data?

Answer: Use KMS for encryption at rest and TLS for transit.

Scenario 25: Your ECS task keeps restarting. Debug?

Answer: Check CloudWatch logs and task memory/CPU limits.

Scenario 26: Application needs global low latency. Solution?

Answer: Use CloudFront CDN.

Scenario 27: How do you expose microservices securely?

Answer: Use API Gateway with JWT/OAuth.

Scenario 28: How do you handle database schema changes?

Answer: Use Flyway or Liquibase with CI/CD.

Scenario 29: High latency between services. Fix?

Answer: Place services in same VPC/AZ and use caching.

Scenario 30: Java batch job runs for hours. Best service?

Answer: Use EC2 or ECS, not Lambda.

Scenario 31: How do you manage Docker secrets?

Answer: Use Secrets Manager with ECS task roles.

Scenario 32: What if SQS message processing fails?

Answer: Use Dead Letter Queue (DLQ).

Scenario 33: How do you ensure message ordering?

Answer: Use FIFO SQS queues.

Scenario 34: Need audit logging of AWS actions?

Answer: Enable CloudTrail.

Scenario 35: How do you rollback deployment?

Answer: Use Blue-Green rollback in CodeDeploy.

Scenario 36: Java app needs scheduled tasks. AWS option?

Answer: Use EventBridge or cron on ECS.

Scenario 37: How do you manage application logs centrally?

Answer: CloudWatch Logs or OpenSearch.

Scenario 38: Need horizontal DB scaling?

Answer: Use Aurora with read replicas.

Scenario 39: How do you protect private EC2 instances?

Answer: Use Bastion host or SSM Session Manager.

Scenario 40: Need HTTPS for app?

Answer: Use ACM with ALB.

Scenario 41: How do you limit outbound internet access?

Answer: Use NAT Gateway.

Scenario 42: How do you manage infrastructure as code?

Answer: Use CloudFormation or Terraform.

Scenario 43: Java app needs object storage. Best choice?

Answer: Amazon S3.

Scenario 44: How do you handle config refresh?

Answer: Reload from Parameter Store at runtime.

Scenario 45: Need high throughput messaging?

Answer: Use Kafka (MSK).

Scenario 46: How do you secure internal service calls?

Answer: Use security groups and IAM roles.

Scenario 47: Application crashes due to memory. Fix?

Answer: Increase JVM heap and ECS memory limits.

Scenario 48: How do you test AWS services locally?

Answer: Use LocalStack or mocks.

Scenario 49: Need file processing pipeline. Design?

Answer: S3 → Lambda → SQS → ECS.

Scenario 50: How do you detect anomalies?

Answer: CloudWatch alarms.

Scenario 51: Java app needs search functionality. AWS service?

Answer: OpenSearch.

Scenario 52: How do you rotate secrets automatically?

Answer: Use Secrets Manager rotation.

Scenario 53: How do you handle multi-region traffic?

Answer: Use Route 53 latency routing.

Scenario 54: Need scalable authentication?

Answer: Use Cognito.

Scenario 55: How do you debug production issues?

Answer: Use logs, metrics, and tracing.

Scenario 56: Need serverless REST API?

Answer: Use API Gateway + Lambda.

Scenario 57: How do you ensure compliance?

Answer: Use IAM policies, CloudTrail, Config.

Scenario 58: How do you shutdown services gracefully?

Answer: Use ALB deregistration delay.

Scenario 59: Java app needs async email sending?

Answer: Use SNS or SES.

Scenario 60: How do you design fault-tolerant systems?

Answer: Use retries, timeouts, circuit breakers.