Senior DevOps Engineer Assessment

Time Limit: 3 hours
Instructions: Answer all questions and provide code/configuration where requested. Focus on demonstrating practical knowledge and best practices.

Section 1: AWS & Cloud Architecture (45 minutes)

Question 1.1: RDS Backup Strategy (15 minutes)

Your company has a production RDS PostgreSQL database in us-east-1 that needs a proper backup and recovery strategy. Current setup only has automated backups enabled with 7-day retention.

Requirements:

- Must be able to restore to any point in the last 30 days
- Need cross-region backup for disaster recovery
- Weekly full backups should be retained for 1 year
- Must minimize cost while meeting requirements

Write the Terraform configuration to implement this backup strategy and explain your approach.

Question 1.2: Application Load Balancer Issues (15 minutes)

Your application behind an ALB is experiencing intermittent 502 errors. The ALB access logs show:

```
2024-01-15T14:30:15.123456Z app/my-alb/1234567890abcdef 192.168.1.100:12345 10.0.1.5:80 0.001 0.002 0.001 502 502 0 123 "GET http://my-app.com/api/users HTTP/1.1" "Mozilla/5.0..." - -
```

Your application servers are running on port 8080 and the health check is configured for `/health`.

List the 5 most likely causes of this issue and provide the AWS CLI commands you would use to investigate each one.

Question 1.3: S3 Bucket Configuration (15 minutes)

Create a complete S3 bucket configuration for a static website hosting setup with the following requirements:

- Serve static content with CloudFront
- Enable versioning with lifecycle policy to move old versions to IA after 30 days
- Block public access but allow CloudFront to access the bucket
- Enable server access logging

Provide the Terraform configuration for the bucket, CloudFront distribution, and necessary IAM policies.

Section 2: Kubernetes & Container Orchestration (45 minutes)

Question 2.1: Fixing a Deployment (20 minutes)

The following deployment is not working correctly. Pods are starting but the application is not accessible:

yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
  namespace: production
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web-app
  template:
    metadata:
      labels:
        app: web-app
    spec:
      containers:
      - name: web
        image: nginx:1.21
        ports:
        - containerPort: 80
        env:
        - name: DATABASE_URL
          value: "postgresql://user:password@localhost:5432/mydb"
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /ready
            port: 8080
          initialDelaySeconds: 5
          periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: web-app-service
  namespace: production
spec:
  selector:
    app: web
  ports:
  - protocol: TCP
```

```
    port: 80
    targetPort: 80
  type: ClusterIP
```

Identify all issues, explain why each is problematic, and provide the corrected configuration.

Question 2.2: Resource Management (25 minutes)

Your production cluster is experiencing resource contention issues. Some pods are being evicted, and others are using more resources than expected.

Create a complete configuration that includes:

1. A deployment for a Node.js API with proper resource requests/limits
2. A HorizontalPodAutoscaler that scales based on CPU and memory
3. A PodDisruptionBudget to ensure availability during updates
4. Proper resource quotas for the namespace

Assume the API typically uses 100Mi memory and 50m CPU under normal load, with spikes up to 500Mi memory and 200m CPU during high traffic.

Section 3: CI/CD & Automation (45 minutes)

Question 3.1: Pipeline Optimization (20 minutes)

Your current GitHub Actions workflow takes 15 minutes to build and test a simple Node.js application. The workflow runs on GitHub-hosted runners and consists of:

yaml

```yaml
name: CI/CD Pipeline
on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
    - name: Install dependencies
      run: npm install
    - name: Build
      run: npm run build
    - name: Upload artifacts
      uses: actions/upload-artifact@v3
      with:
        name: dist
        path: dist/

  test:
    runs-on: ubuntu-latest
    needs: build
    steps:
    - uses: actions/checkout@v3
    - name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
    - name: Install dependencies
      run: npm install
    - name: Run tests
      run: npm run test
    - name: Run linter
      run: npm run lint

  deploy:
    runs-on: ubuntu-latest
    needs: [build, test]
    if: github.ref == 'refs/heads/main'
```

```
steps:
- uses: actions/checkout@v3
- name: Download artifacts
  uses: actions/download-artifact@v3
  with:
    name: dist
    path: dist/
- name: Deploy to K8s
  run: kubectl apply -f k8s/
```

Optimize this workflow to reduce build time to under 5 minutes while maintaining quality. Consider caching, parallelization, and any other improvements.

Question 3.2: Deployment Strategy (25 minutes)

Design a safe deployment strategy for a REST API with the following requirements:

- Zero-downtime deployments
- Automatic rollback if health checks fail
- Gradual traffic shifting (10% → 50% → 100%)
- Easy manual rollback option
- Support for feature flags

Choose either GitHub Actions or Jenkins and provide a complete pipeline that implements this strategy using Kubernetes. Include the necessary Kubernetes manifests (Service, Deployment, etc.).

Section 4: Security & Compliance (45 minutes)

Question 4.1: Security Audit (20 minutes)

You're conducting a security audit and find the following configuration in your EKS cluster:

yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: developers-binding
subjects:
- kind: Group
  name: developers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

And this pod security configuration:

yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-app
spec:
  securityContext:
    runAsUser: 0
    runAsGroup: 0
  containers:
  - name: app
    image: myapp:latest
    securityContext:
      privileged: true
      allowPrivilegeEsculation: true
    volumeMounts:
    - name: docker-socket
      mountPath: /var/run/docker.sock
  volumes:
  - name: docker-socket
    hostPath:
      path: /var/run/docker.sock
```

Identify the security issues and provide secure alternatives for each problematic configuration.

Question 4.2: Secrets Management (25 minutes)

Design a secure secrets management strategy for a microservices application with the following requirements:

- Secrets must be encrypted at rest and in transit
- Application pods should not have access to secrets they don't need
- Secrets rotation should be automated
- Integration with external secret management (like AWS Secrets Manager)
- Audit logging of secret access

Provide the implementation using Kubernetes native tools and one external solution. Include the necessary RBAC configuration and example usage in a deployment.

Bonus Section: System Design (15 minutes)

Bonus Question: Simple Architecture

Design a simple but scalable architecture for a web application that allows users to upload and resize images. Requirements:

- Handle 1000 uploads per day (with 3x spikes)
- Resize images to 3 different sizes (thumbnail, medium, large)
- Store original and resized images with 99.9% durability
- Serve images with low latency globally
- Cost-optimized for variable workloads

Draw a simple architecture diagram (or describe it) and explain your choice of AWS services.

Evaluation Criteria

Technical Accuracy (40%):

- Correct solution to the problem
- Understanding of key concepts
- Proper use of tools and services

Best Practices (30%):

- Security considerations
- Scalability and reliability
- Code quality and maintainability

Problem-Solving (20%):

- Logical approach to debugging

- Consideration of edge cases
- Practical solutions

Communication (10%):

- Clear explanation of reasoning
- Proper documentation
- Structured approach

Time Management: If you cannot complete all sections, prioritize demonstrating understanding over rushing through all questions. Quality over quantity.

End of Assessment