SDLC AutomationDomain 1 SDLC Automation (22% of scored content)

Weight: 22%

- Sample DOP-C02 Scenarios (Domain 1)
  - Automatically deploy an application only after it passes security scans: Use CodePipeline with a test stage that runs CodeBuild + scanner before deploy.
  - Deploy a Lambda version with a 10% traffic test: Use CodeDeploy canary deployment with Lambda alias traffic shifting.
  - Detect manually changed infrastructure: Use CloudFormation drift detection.
  - Roll back a failed deployment to ECS: Use CodeDeploy with blue/green + auto rollback.
  - Deploy IaC across accounts: Use CloudFormation StackSets with org-level permissions.
- Domain 1 Summary Cheat Sheet
  - Source control: CodeCommit
  - Build/test: CodeBuild
  - CI/CD orchestration: CodePipeline
  - Deployment: CodeDeploy, SAM, Beanstalk
  - IaC: CloudFormation, CDK, Terraform
  - Testing: Unit, integration, security in CodeBuild
  - Versioning: CodeArtifact
  - Rollback: CodeDeploy hooks, CFN auto rollback
  - Multi-account deploy: StackSets
- Recommended AWS Reading
  - CI/CD on AWS Whitepaper
  - AWS CodePipeline User Guide
  - AWS DevOps Blog
  - CloudFormation Best Practices
  - CodeArtifact Overview

Core Competency

Design and implement automated software development lifecycle (SDLC) processes including CI/CD pipelines, version control, infrastructure automation, and testing using AWS services.

1.1 Design, Implement, and Manage Continuous Integration and Continuous Delivery (CI/CD) Systems

- Core AWS Services for CI/CD
  - CodeCommit: Git-based repository
  - CodeBuild: Build and test automation
  - CodeDeploy: Automate deployments to EC2, Lambda, ECS
  - CodePipeline: CI/CD orchestration
  - CodeArtifact: Artifact/package repository
  - CloudFormation / CDK / SAM: Infrastructure as Code tools
  - EventBridge: Event-driven pipeline triggers
  - Elastic Beanstalk: PaaS with integrated deployment lifecycle
- CI/CD Pipeline Components
  - Source: CodeCommit, GitHub, S3
  - Build: CodeBuild, Jenkins, Maven, Gradle
  - Test: Unit/integration tests in CodeBuild or 3rd party
  - Deploy: CodeDeploy, CloudFormation, SAM, Beanstalk, ECS
  - Approval: Manual approval action in CodePipeline
  - Notification: SNS, EventBridge, ChatOps integration (e.g., Slack)
  - Use parallel actions in CodePipeline to speed up multi-region or multi-stage builds.
- CodePipeline Deep Dive
  - Artifacts: Output from one stage, input to next
  - Manual approvals: Insert gate before production
  - Event triggers: Code commit, tag, or CloudWatch event
  - Custom actions: Invoke Lambda or external integration
  - Ensure each stage is idempotent, and include rollback strategies.

1.3 Implement and Manage Infrastructure as Code (IaC)

- IaC Tools on AWS
  - CloudFormation: YAML/JSON, Declarative, AWS-native
  - CDK: TypeScript, Python, Java, Imperative; compiles to CFN
  - SAM: YAML, Focused on serverless apps
  - Terraform: HCL, 3rd-party, supports multiple providers
  - Use StackSets for deploying templates across multiple AWS accounts or regions.
- CloudFormation Key Concepts
  - Stacks: Collection of AWS resources managed as a unit
  - Nested Stacks: Modular templates
  - Change Sets: Preview impact of updates
  - Stack Policies: Restrict resource updates (e.g., don't replace DB)
  - Drift Detection: Alert when manual changes occur outside IaC
- Automating IaC in Pipelines
  - Include IaC validation as part of build stage
  - Use cfn-lint, cfn-nag or CDK synth/test
  - Deploy using:
  - aws cloudformation deploy (CLI)
  - sam deploy
  - cdk deploy
  - Rollbacks triggered automatically if failure occurs

1.5 Integrate Version Control and Artifact Management

- CodeCommit Essentials
  - Git support: Standard Git commands work
  - Encryption: At rest (KMS), in transit (HTTPS)
  - Triggers: Lambda, SNS, or CodePipeline integration
  - IAM: Fine-grained access to branches/repos
  - Pull Requests: Integrated review and merge workflows
  - Supports both SSH and HTTPS, and integrates natively with IAM roles.
- CodeArtifact Overview
  - Package formats: npm, Maven, PyPI, NuGet
  - Upstream repos: Pull from public repos like npmjs
  - Access control: IAM or repository policies
  - Retention policies: Control storage and cost of old artifacts
  - Use in CI/CD to store versioned packages, manage license/security risk.

1.4 Automate Deployment Strategies and Rollbacks

- Deployment Strategies
  - All-at-once: Fast, no rollback support
  - Rolling: Replace instances in batches
  - Canary: Deploy to subset, monitor, then full rollout
  - Blue/Green: Create new environment, switch traffic when stable
  - Traffic shifting: Gradual rollout (Lambda/ALB with weights)
  - Use CodeDeploy with AppSpec.yml and hooks (BeforeInstall, AfterInstall, etc.)
- Rollback Mechanisms
  - CloudFormation auto-rollback on failure
  - CodeDeploy supports rollback hooks
  - Pipeline stage retries and alerts via SNS/EventBridge
  - Always test rollback paths during staging deployments.

1.2 Automate Testing and Quality Checks

- Test Automation in CI/CD
  - Unit Tests: Validate logic (CodeBuild phase)
  - Integration Tests: Run against deployed service or DB
  - Security Scans: Scan packages (e.g., using CodeGuru, SonarQube)
  - Load Tests: Use tools like JMeter or Artillery
  - Static Code Analysis: Find code smells and security flaws
  - Add testing stages after build and pre-deploy in CodePipeline.
- AWS Tools for Quality
  - CodeGuru Reviewer: Detects security and code quality issues in pull requests
  - CodeBuild Reports: Integrates test results, code coverage
  - Inspector / Macie: Detect vulnerabilities in deployment artifacts
  - CloudWatch Logs/Insights: Real-time log analysis post-deployment

Presented with xmind AI