aws training and certification

AWS Certified DevOps Engineer – Professional (DOP-C01) Exam Guide

Introduction

The AWS Certified DevOps Engineer – Professional (DOP-C01) exam is intended for individuals who perform a DevOps engineer role. The exam validates a candidate's technical expertise in provisioning, operating, and managing distributed application systems on the AWS platform.

The exam also validates a candidate's ability to complete the following tasks:

- Implement and manage continuous delivery systems and methodologies on AWS
- Implement and automate security controls, governance processes, and compliance validation
- Define and deploy monitoring, metrics, and logging systems on AWS
- Implement systems that are highly available, scalable, and self-healing on the AWS platform
- Design, manage, and maintain tools to automate operational processes

Target candidate description

The target candidate is expected to have 2 or more years of experience provisioning, operating, and managing AWS environments. The target candidate must also have experience developing code in at least one high-level programming language.

Recommended AWS knowledge

The target candidate should have the following knowledge:

- Experience building highly automated infrastructures
- Experience administering operating systems
- An understanding of modern development and operations processes and methodologies

What is considered out of scope for the target candidate?

The following is a non-exhaustive list of related job tasks that the target candidate is not expected to be able to perform. These items are considered out of scope for the exam:

- Advanced networking (advanced routing algorithms, failover techniques)
- Deep-level security recommendations for developers
- Database query and performance optimization
- Full-stack application code development
- Normalization of data schemes

For a detailed list of specific tools and technologies that might be covered on the exam, as well as lists of in-scope AWS services, refer to the Appendix.

Exam content

Response types

There are two types of questions on the exam:

- **Multiple choice:** Has one correct response and three incorrect responses (distractors)
- **Multiple response:** Has two or more correct responses out of five or more response options

Select one or more responses that best complete the statement or answer the question. Distractors, or incorrect answers, are response options that a candidate with incomplete knowledge or skill might choose. Distractors are generally plausible responses that match the content area.

Unanswered questions are scored as incorrect; there is no penalty for guessing. The exam includes 65 questions that will affect your score.

Unscored content

The exam includes 10 unscored questions that do not affect your score. AWS collects information about candidate performance on these unscored questions to evaluate these questions for future use as scored questions. These unscored questions are not identified on the exam.

Exam results

The AWS Certified DevOps Engineer – Professional (DOP-C01) exam is a pass or fail exam. The exam is scored against a minimum standard established by AWS professionals who follow certification industry best practices and guidelines.

Your results for the exam are reported as a scaled score of 100–1,000. The minimum passing score is 750. Your score shows how you performed on the exam as a whole and whether or not you passed. Scaled scoring models help equate scores across multiple exam forms that might have slightly different difficulty levels.

Your score report may contain a table of classifications of your performance at each section level. This information is intended to provide general feedback about your exam performance. The exam uses a compensatory scoring model, which means that you do not need to achieve a passing score in each section. You need to pass only the overall exam.

Each section of the exam has a specific weighting, so some sections have more questions than other sections have. The table contains general information that highlights your strengths and weaknesses. Use caution when interpreting section-level feedback.

Content outline

This exam guide includes weightings, test domains, and objectives for the exam. It is not a comprehensive listing of the content on the exam. However, additional context for each of the objectives is available to help guide your preparation for the exam. The following table lists the main content domains and their weightings. The table precedes the complete exam content outline, which includes the additional context. The percentage in each domain represents only scored content.

Domain	% of Exam
Domain 1: SDLC Automation	22%
Domain 2: Configuration Management and Infrastructure as Code	19%
Domain 3: Monitoring and Logging	15%
Domain 4: Policies and Standards Automation	10%
Domain 5: Incident and Event Response	18%
Domain 6: High Availability, Fault Tolerance, and Disaster Recover	16%
TOTAL	**100%**

Domain 1: SDLC Automation

1.1 Apply concepts required to automate a CI/CD pipeline
- Set up repositories
- Set up build services
- Integrate automated testing (e.g., unit tests, integrity tests)
- Set up deployment products/services
- Orchestrate multiple pipeline stages

1.2 Determine source control strategies and how to implement them
- Determine a workflow for integrating code changes from multiple contributors
- Assess security requirements and recommend code repository access design
- Reconcile running application versions to repository versions (tags)
- Differentiate different source control types

1.3 Apply concepts required to automate and integrate testing
- Run integration tests as part of code merge process
- Run load/stress testing and benchmark applications at scale
- Measure application health based on application exit codes (robust Health Check)
- Automate unit tests to check pass/fail, code coverage
  - CodePipeline, CodeBuild, etc.
- Integrate tests with pipeline

1.4 Apply concepts required to build and manage artifacts securely
- Distinguish storage options based on artifacts security classification
- Translate application requirements into Operating System and package configuration (build specs)
- Determine the code/environment dependencies and required resources
  - Example: CodeDeploy AppSpec, CodeBuild buildspec
- Run a code build process

1.5 Determine deployment/delivery strategies (e.g., A/B, Blue/green, Canary, Red/black) and how to implement them using AWS services
- Determine the correct delivery strategy based on business needs
- Critique existing deployment strategies and suggest improvements
- Recommend DNS/routing strategies (e.g., Route 53, ELB, ALB, load balancer) based on business continuity goals
- Verify deployment success/failure and automate rollbacks

Domain 2: Configuration Management and Infrastructure as Code

2.1 Determine deployment services based on deployment needs
- Demonstrate knowledge of process flows of deployment models
- Given a specific deployment model, classify and implement relevant AWS services to meet requirements
    - Given the requirement to have DynamoDB choose CloudFormation instead of OpsWorks
    - Determine what to do with rolling updates

2.2 Determine application and infrastructure deployment models based on business needs
- Balance different considerations (cost, availability, time to recovery) based on business requirements to choose the best deployment model
- Determine a deployment model given specific AWS services
- Analyze risks associated with deployment models and relevant remedies

2.3 Apply security concepts in the automation of resource provisioning
- Choose the best automation tool given requirements
- Demonstrate knowledge of security best practices for resource provisioning (e.g., encrypting data bags, generating credentials on the fly)
- Review IAM policies and assess if sufficient but least privilege is granted for all lifecycle stages of a deployment (e.g., create, update, promote)
- Review credential management solutions (e.g., EC2 parameter store, third party)
- Build the automation
    - CloudFormation template, Chef Recipe, Cookbooks, Code pipeline, etc.

2.4 Determine how to implement lifecycle hooks on a deployment
- Determine appropriate integration techniques to meet project requirements
- Choose the appropriate hook solution (e.g., implement leader node selection after a node failure) in an Auto Scaling group
- Evaluate hook implementation for failure impacts (if a remote call fails, if a dependent service is temporarily unavailable (i.e., Amazon S3), and recommend resiliency improvements
- Evaluate deployment rollout procedures for failure impacts and evaluate rollback/recovery processes

2.5 Apply concepts required to manage systems using AWS configuration management tools and services
- Identify pros and cons of AWS configuration management tools
- Demonstrate knowledge of configuration management components
- Show the ability to run configuration management services end to end with no assistance while adhering to industry best practices

Domain 3: Monitoring and Logging

3.1 Determine how to set up the aggregation, storage, and analysis of logs and metrics

- Implement and configure distributed logs collection and processing (e.g., agents, syslog, flumed, CW agent)
- Aggregate logs (e.g., Amazon S3, CW Logs, intermediate systems (EMR), Kinesis FH – Transformation, ELK/BI)
- Implement custom CW metrics, Log subscription filters
- Manage Log storage lifecycle (e.g., CW to S3, S3 lifecycle, S3 events)

3.2 Apply concepts required to automate monitoring and event management of an environment

- Parse logs (e.g., Amazon S3 data events/event logs/ELB/ALB/CF access logs) and correlate with other alarms/events (e.g., CW events to AWS Lambda) and take appropriate action
- Use CloudTrail/VPC flow logs for detective control (e.g., CT, CW log filters, Athena, NACL or WAF rules) and take dependent actions (AWS step) based on error handling logic (state machine)
- Configure and implement Patch/inventory/state management using ESM (SSM), Inspector, CodeDeploy, OpsWorks, and CW agents
    - EC2 retirement/maintenance
- Handle scaling/failover events (e.g., ASG, DB HA, route table/DNS update, Application Config, Auto Recovery, PH dashboard, TA)
- Determine how to automate the creation of monitoring

3.3 Apply concepts required to audit, log, and monitor operating systems, infrastructures, and applications

- Monitor end to end service metrics (DDB/S3) using available AWS tools (X-ray with EB and Lambda)
- Verify environment/OS state through auditing (Inspector), Config rules, CloudTrail (process and action), and AWS APIs
- Enable, configure, and analyze custom metrics (e.g., Application metrics, memory, KCL/KPL) and take action
- Ensure container monitoring (e.g., task state, placement, logging, port mapping, LB)
- Distinguish between services that enable service level or OS level monitoring
    - Example: AWS services that use OS agents (e.g., Inspector, SSM)

3.4 Determine how to implement tagging and other metadata strategies

- Segregate authority based on tagging (lifecycle stages – dev/prod) with Condition context keys
- Utilize Amazon S3 system/user-defined metadata for classification and automation
- Design and implement tag-based deployment groups with CodeDeploy
- Best practice for cost allocation/optimization with tagging

Domain 4: Policies and Standards Automation

4.1 Apply concepts required to enforce standards for logging, metrics, monitoring, testing, and security

- Detect, report, and respond to governance and security violations
- Apply logging standards across application, operating system, and infrastructure
- Apply context specific application health and performance monitoring
- Outline standards for delivery models for logs and metrics (e.g., JSON, XML, Data Normalization)

4.2 Determine how to optimize cost through automation

- Prioritize automation effort to reduce labor costs
- Implement right sizing of workload based on metrics
- Assess ways to improve time to market through automating process orchestration and repeatable tasks
- Diagnose outliers to determine use case fit
    - Example: Configuration drift
- Measure and automate cost optimization through events
    - Example: Trusted Advisor

4.3 Apply concepts required to implement governance strategies

- Generalize governance standards across CI/CD pipeline
- Outline and measure the real-time status of compliance with governance strategies
- Report on compliance with governance strategies
- Deploy governance policies related to self-service capabilities
    - Example: Service Catalog, CFN Nag

Domain 5: Incident and Event Response

5.1 Troubleshoot issues and determine how to restore operations

- Given an issue, evaluate how to narrow down the unhealthy components as quickly as possible
- Given an increase in load, determine what steps to take to mitigate the impact
- Determine the causes and impacts of a failure
    - Example: Deployment, operations
- Determine the best way to restore operations after a failure occurs
- Investigate and correlate logged events with application components
    - Example: application source code

5.2 Determine how to automate event management and alerting

- Set up automated restores from backup in the event of a catastrophic failure
- Set up methods to deliver alerts and notifications that are appropriate for different types of events
- Assess the quality/actionability of alerts
- Configure metrics appropriate to an application's SLAs
- Proactively update limits

5.3 Apply concepts required to implement automated healing

- Set up the correct scaling strategy to enable auto-healing when a failure occurs (e.g., with Auto Scaling policies)
- Use the correct rollback strategy to avoid impact from failed deployments
- Configure Route 53 to ensure cross-Region failover
- Detect and respond to maintenance or Spot termination events

5.4 Apply concepts required to set up event-driven automated actions

- Configure Lambda functions or CloudWatch actions to implement automated actions
- Set up CloudWatch event rules and/or Config rules and targets
- Use AWS Systems Manager or Step Functions to coordinate components (e.g., Lambda, use maintenance windows)
- Configure a build/roll-out process to automatically respond to critical software updates

Domain 6: High Availability, Fault Tolerance, and Disaster Recovery

6.1 Determine appropriate use of multi-AZ versus multi-Region architectures

- Determine deployment strategy based on HA/DR requirements
- Determine data replication strategy based on cost and durability requirements
- Determine infrastructure, platform, and services based on HA/DR requirements
- Design for HA/FT/DR based on service availability (i.e., global/regional/single AZ)

6.2 Determine how to implement high availability, scalability, and fault tolerance

- Design deployment strategy to support HA/FT/scalability
- Assess statefulness of application infrastructure components
- Use load balancing to distribute traffic across multiple AZ/ASGs/instance types (spot/M4 vs C4) /targets
- Use appropriate caching solutions to improve availability and performance

6.3 Determine the right services based on business needs (e.g., RTO/RPO, cost)

- Determine cost-effective storage solution for your application
  - Example: tiered, archival, EBS type, hot/cold
- Choose a database platform and configuration to meet business requirements
- Choose a cost-effective compute platform based on business requirements
  - Example: Spot
- Choose a deployment service/model based on business requirements
  - Example: Code Deploy, Blue/Green deployment
- Determine when to use managed service vs. self-managed infrastructure (Docker on EC2 vs. ECS)

6.4 Determine how to design and automate disaster recovery strategies

- Automate failure detection
- Automate components/environment recovery
- Choose appropriate deployment strategy for environment recovery
- Design automation to support failover in hybrid environment

6.5 Evaluate a deployment for points of failure

- Determine appropriate deployment-specific health checks
- Implement failure detection during deployment
- Implement failure event handling/response
- Ensure that resources/components/processes exist to react to failures during deployment
- Look for exit codes on each event of the deployment
- Map errors to different points of deployment

Appendix

Which key tools, technologies, and concepts might be covered on the exam?

The following is a non-exhaustive list of the tools and technologies that could appear on the exam. This list is subject to change and is provided to help you understand the general scope of services, features, or technologies on the exam. The general tools and technologies in this list appear in no particular order. AWS services are grouped according to their primary functions. While some of these technologies will likely be covered more than others on the exam, the order and placement of them in this list is no indication of relative weight or importance:

- Application deployment
- Application integration
- Application pipelines
- Automation
- Code repository best practices
- Cost optimization
- Deployment requirements
- Hybrid deployments
- IAM policies
- Metrics, monitoring, alarms, and logging
- Network ACL and security group design and implementation
- Operational best practices
- Rollback procedures

AWS services and features

Analytics:

- Amazon Athena
- Amazon EMR
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- Amazon QuickSight

Compute:

- Amazon EC2
- Amazon EC2 Auto Scaling

Containers:

- AWS App Runner
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- AWS Fargate

Database:

- Amazon DynamoDB
- Amazon RDS
- Amazon Redshift

Developer Tools:

- AWS Cloud Development Kit (AWS CDK)
- AWS CloudShell
- AWS CodeArtifact
- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- Amazon CodeGuru
- AWS CodePipeline
- AWS CodeStar
- AWS Command Line Interface (CLI)
- AWS X-Ray

Management and Governance:

- AWS CloudFormation
- AWS CloudTrail
- Amazon CloudWatch
- AWS Config
- AWS OpsWorks
- AWS Organizations
- AWS Systems Manager
- AWS Trusted Advisor

Networking and Content Delivery:

- Amazon API Gateway
- AWS Client VPN
- Amazon CloudFront
- Amazon Route 53
- AWS Site-to-Site VPN
- AWS Transit Gateway
- Amazon VPC
- Elastic Load Balancing

Security, Identity, and Compliance:

- Amazon GuardDuty
- AWS Identity and Access Management (IAM)
- Amazon Inspector
- AWS Key Management Service (AWS KMS)
- AWS Secrets Manager
- AWS Single Sign-On
- AWS WAF

Serverless:

- Amazon EventBridge (Amazon CloudWatch Events)
- AWS Lambda
- AWS Serverless Application Model (AWS SAM)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Step Functions

Storage:

- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic File System (Amazon EFS)
- Amazon S3
- AWS Storage Gateway