aws training and certification

AWS Certified Developer – Associate (DVA-C01) Exam Guide

Introduction

The AWS Certified Developer – Associate (DVA-C01) exam is intended for individuals who perform a developer role. The exam validates a candidate's ability to do the following:

- Demonstrate an understanding of core AWS services, uses, and basic AWS architecture best practices
- Demonstrate proficiency in developing, deploying, and debugging cloud-based applications by using AWS

Target candidate description

The target candidate should have 1 or more years of hands-on experience developing and maintaining an AWS-based application.

Recommended general IT knowledge

The target candidate should have the following:

- In-depth knowledge of at least one high-level programming language
- Understanding of application lifecycle management
- The ability to write code for serverless applications
- Understanding of the use of containers in the development process

Recommended AWS knowledge

The target candidate should be able to do the following:

- Use the AWS service APIs, CLI, and software development kits (SDKs) to write applications
- Identify key features of AWS services
- Understand the AWS shared responsibility model
- Use a continuous integration and continuous delivery (CI/CD) pipeline to deploy applications on AWS
- Use and interact with AWS services
- Apply basic understanding of cloud-native applications to write code
- Write code by using AWS security best practices (for example, use IAM roles instead of secret and access keys in the code)
- Author, maintain, and debug code modules on AWS

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:

- Design architectures (for example, distributed system, microservices)
- Design and implement CI/CD pipelines

- Administer IAM users and groups
- Administer Amazon Elastic Container Service (Amazon ECS)
- Design AWS networking infrastructure (for example, Amazon VPC, AWS Direct Connect)
- Understand compliance and licensing

For a detailed list of specific tools and technologies that might be covered on the exam, as well as lists of in-scope and out-of-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 50 questions that will affect your score.

Unscored content

The exam includes 15 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 Developer – Associate (DVA-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 720. Your score shows how you performed on the exam as a whole and whether you passed. Scaled scoring models help equate scores across multiple exam forms that might have slightly different difficulty levels.

Your score report could 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: Deployment	22%
Domain 2: Security	26%
Domain 3: Development with AWS Services	30%
Domain 4: Refactoring	10%
Domain 5: Monitoring and Troubleshooting	12%
TOTAL	**100%**

Domain 1: Deployment

1.1 Deploy written code in AWS using existing CI/CD pipelines, processes, and patterns.
- Commit code to a repository and invoke build, test and/or deployment actions
- Use labels and branches for version and release management
- Use AWS CodePipeline to orchestrate workflows against different environments
- Apply AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeStar, and AWS CodeDeploy for CI/CD purposes
- Perform a roll back plan based on application deployment policy

1.2 Deploy applications using AWS Elastic Beanstalk.
- Utilize existing supported environments to define a new application stack
- Package the application
- Introduce a new application version into the Elastic Beanstalk environment
- Utilize a deployment policy to deploy an application version (i.e., all at once, rolling, rolling with batch, immutable)
- Validate application health using Elastic Beanstalk dashboard
- Use Amazon CloudWatch Logs to instrument application logging

1.3 Prepare the application deployment package to be deployed to AWS.
- Manage the dependencies of the code module (like environment variables, config files and static image files) within the package
- Outline the package/container directory structure and organize files appropriately
- Translate application resource requirements to AWS infrastructure parameters (e.g., memory, cores)

1.4 Deploy serverless applications.
- Given a use case, implement and launch an AWS Serverless Application Model (AWS SAM) template
- Manage environments in individual AWS services (e.g., Differentiate between Development, Test, and Production in Amazon API Gateway)

Domain 2: Security

2.1 Make authenticated calls to AWS services.
- Communicate required policy based on least privileges required by application.
- Assume an IAM role to access a service
- Use the software development kit (SDK) credential provider on-premises or in the cloud to access AWS services (local credentials vs. instance roles)

2.2 Implement encryption using AWS services.
- Encrypt data at rest (client side; server side; envelope encryption) using AWS services
- Encrypt data in transit

2.3 Implement application authentication and authorization.
- Add user sign-up and sign-in functionality for applications with Amazon Cognito identity or user pools
- Use Amazon Cognito-provided credentials to write code that access AWS services.
- Use Amazon Cognito sync to synchronize user profiles and data
- Use developer-authenticated identities to interact between end user devices, backend authentication, and Amazon Cognito

Domain 3: Development with AWS Services

3.1 Write code for serverless applications.
- Compare and contrast server-based vs. serverless model (e.g., micro services, stateless nature of serverless applications, scaling serverless applications, and decoupling layers of serverless applications)
- Configure AWS Lambda functions by defining environment variables and parameters (e.g., memory, time out, runtime, handler)
- Create an API endpoint using Amazon API Gateway
- Create and test appropriate API actions like GET, POST using the API endpoint
- Apply Amazon DynamoDB concepts (e.g., tables, items, and attributes)
- Compute read/write capacity units for Amazon DynamoDB based on application requirements
- Associate an AWS Lambda function with an AWS event source (e.g., Amazon API Gateway, Amazon CloudWatch event, Amazon S3 events, Amazon Kinesis)
- Invoke an AWS Lambda function synchronously and asynchronously

3.2 Translate functional requirements into application design.
- Determine real-time vs. batch processing for a given use case
- Determine use of synchronous vs. asynchronous for a given use case
- Determine use of event vs. schedule/poll for a given use case
- Account for tradeoffs for consistency models in an application design

3.3 Implement application design into application code.

- Write code to utilize messaging services (e.g., SQS, SNS)
- Use Amazon ElastiCache to create a database cache
- Use Amazon DynamoDB to index objects in Amazon S3
- Write a stateless AWS Lambda function
- Write a web application with stateless web servers (Externalize state)

3.4 Write code that interacts with AWS services by using APIs, SDKs, and AWS CLI.

- Choose the appropriate APIs, software development kits (SDKs), and CLI commands for the code components
- Write resilient code that deals with failures or exceptions (i.e., retries with exponential back off and jitter)

Domain 4: Refactoring

4.1 Optimize applications to best use AWS services and features.

- Implement AWS caching services to optimize performance (e.g., Amazon ElastiCache, Amazon API Gateway cache)
- Apply an Amazon S3 naming scheme for optimal read performance

4.2 Migrate existing application code to run on AWS.

- Isolate dependencies
- Run the application as one or more stateless processes
- Develop in order to enable horizontal scalability
- Externalize state

Domain 5: Monitoring and Troubleshooting

5.1 Write code that can be monitored.

- Create custom Amazon CloudWatch metrics
- Perform logging in a manner available to systems operators
- Instrument application source code to enable tracing in AWS X-Ray

5.2 Perform root cause analysis on faults found in testing or production.

- Interpret the outputs from the logging mechanism in AWS to identify errors in logs
- Check build and testing history in AWS services (e.g., AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline) to identify issues
- Utilize AWS services (e.g., Amazon CloudWatch, VPC Flow Logs, and AWS X-Ray) to locate a specific faulty component

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:

- Analytics
- Application Integration
- Containers
- Cost and Capacity Management
- Data Movement
- Developer Tools
- Instances (virtual machines)
- Management and Governance
- Networking and Content Delivery
- Security
- Serverless

AWS services and features

Analytics:

- Amazon Elasticsearch Service (Amazon ES)
- Amazon Kinesis

Application Integration:

- Amazon EventBridge (Amazon CloudWatch Events)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Step Functions

Compute:

- Amazon EC2
- AWS Elastic Beanstalk
- AWS Lambda

Containers:

- Amazon Elastic Container Registry (Amazon ECR)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Services (Amazon EKS)

Database:

- Amazon DynamoDB
- Amazon ElastiCache
- Amazon RDS

Developer Tools:

- AWS CodeArtifact
- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- Amazon CodeGuru
- AWS CodePipeline
- AWS CodeStar
- AWS Fault Injection Simulator
- AWS X-Ray

Management and Governance:

- AWS CloudFormation
- Amazon CloudWatch

Networking and Content Delivery:

- Amazon API Gateway
- Amazon CloudFront
- Elastic Load Balancing

Security, Identity, and Compliance:

- Amazon Cognito
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)

Storage:

- Amazon S3

Out-of-scope AWS services and features

The following is a non-exhaustive list of AWS services and features that are not covered on the exam. These services and features do not represent every AWS offering that is excluded from the exam content. Services or features that are entirely unrelated to the target job roles for the exam are excluded from this list because they are assumed to be irrelevant.

Out-of-scope AWS services and features include the following:

- AWS Application Discovery Service
- Amazon AppStream 2.0
- Amazon Chime
- Amazon Connect
- AWS Database Migration Service (AWS DMS)
- AWS Device Farm
- Amazon Elastic Transcoder
- Amazon GameLift
- Amazon Lex
- Amazon Machine Learning (Amazon ML)
- AWS Managed Services
- Amazon Mobile Analytics
- Amazon Polly

- Amazon QuickSight
- Amazon Rekognition
- AWS Server Migration Service (AWS SMS)
- AWS Service Catalog
- AWS Shield Advanced
- AWS Shield Standard
- AWS Snow Family
- AWS Storage Gateway
- AWS WAF
- Amazon WorkMail
- Amazon WorkSpaces

1) A company is migrating a legacy application to Amazon EC2. The application uses a user name and password stored in the source code to connect to a MySQL database. The database will be migrated to an Amazon RDS for MySQL DB instance. As part of the migration, the company wants to implement a secure way to store and automatically rotate the database credentials.

Which approach meets these requirements?

A) Store the database credentials in environment variables in an Amazon Machine Image (AMI). Rotate the credentials by replacing the AMI.
B) Store the database credentials in AWS Systems Manager Parameter Store. Configure Parameter Store to automatically rotate the credentials.
C) Store the database credentials in environment variables on the EC2 instances. Rotate the credentials by relaunching the EC2 instances.
D) Store the database credentials in AWS Secrets Manager. Configure Secrets Manager to automatically rotate the credentials.

2) A developer is designing a web application that allows the users to post comments and receive near-real-time feedback.

Which architectures meet these requirements? (Select TWO.)

A) Create an AWS AppSync schema and corresponding APIs. Use an Amazon DynamoDB table as the data store.
B) Create a WebSocket API in Amazon API Gateway. Use an AWS Lambda function as the backend and an Amazon DynamoDB table as the data store.
C) Create an AWS Elastic Beanstalk application backed by an Amazon RDS database. Configure the application to allow long-lived TCP/IP sockets.
D) Create a GraphQL endpoint in Amazon API Gateway. Use an Amazon DynamoDB table as the data store.
E) Enable WebSocket on Amazon CloudFront. Use an AWS Lambda function as the origin and an Amazon Aurora DB cluster as the data store.

3) A developer is adding sign-up and sign-in functionality to an application. The application is required to make an API call to a custom analytics solution to log user sign-in events.

Which combination of actions should the developer take to satisfy these requirements? (Select TWO.)

A) Use Amazon Cognito to provide the sign-up and sign-in functionality.
B) Use AWS IAM to provide the sign-up and sign-in functionality.
C) Configure an AWS Config rule to make the API call triggered by the post-authentication event.
D) Invoke an Amazon API Gateway method to make the API call triggered by the post-authentication event.
E) Execute an AWS Lambda function to make the API call triggered by the post-authentication event.

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved | aws.amazon.com

4) A company is using Amazon API Gateway for its REST APIs in an AWS account. The security team wants to allow only IAM users from another AWS account to access the APIs.

Which combination of actions should the security team take to satisfy these requirements? (Select TWO.)

A) Create an IAM permission policy and attach it to each IAM user. Set the APIs method authorization type to AWS_IAM. Use Signature Version 4 to sign the API requests.
B) Create an Amazon Cognito user pool and add each IAM user to the pool. Set the method authorization type for the APIs to COGNITO_USER_POOLS. Authenticate using the IAM credentials in Amazon Cognito and add the ID token to the request headers.
C) Create an Amazon Cognito identity pool and add each IAM user to the pool. Set the method authorization type for the APIs to COGNITO_USER_POOLS. Authenticate using the IAM credentials in Amazon Cognito and add the access token to the request headers.
D) Create a resource policy for the APIs that allows access for each IAM user only.
E) Create an Amazon Cognito authorizer for the APIs that allows access for each IAM user only. Set the method authorization type for the APIs to COGNITO_USER_POOLS.

5) A developer is building an application that transforms text files to .pdf files. The text files are written to a source Amazon S3 bucket by a separate application. The developer wants to read the files as they arrive in Amazon S3 and convert them to .pdf files using AWS Lambda. The developer has written an IAM policy to allow access to Amazon S3 and Amazon CloudWatch Logs.

Which actions should the developer take to ensure that the Lambda function has the correct permissions?

A) Create a Lambda execution role using AWS IAM. Attach the IAM policy to the role. Assign the Lambda execution role to the Lambda function.
B) Create a Lambda execution user using AWS IAM. Attach the IAM policy to the user. Assign the Lambda execution user to the Lambda function.
C) Create a Lambda execution role using AWS IAM. Attach the IAM policy to the role. Store the IAM role as an environment variable in the Lambda function.
D) Create a Lambda execution user using AWS IAM. Attach the IAM policy to the user. Store the IAM user credentials as environment variables in the Lambda function.

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved | aws.amazon.com

6) A company has AWS workloads in multiple geographical locations. A developer has created an Amazon Aurora database in the us-west-1 Region. The database is encrypted using a customer-managed AWS KMS key. Now the developer wants to create the same encrypted database in the us-east-1 Region.

Which approach should the developer take to accomplish this task?

A) Create a snapshot of the database in the us-west-1 Region. Copy the snapshot to the us-east-1 Region and specify a KMS key in the us-east-1 Region. Restore the database from the copied snapshot.
B) Create an unencrypted snapshot of the database in the us-west-1 Region. Copy the snapshot to the us-east-1 Region. Restore the database from the copied snapshot and enable encryption using the KMS key from the us-east-1 Region.
C) Disable encryption on the database. Create a snapshot of the database in the us-west-1 Region. Copy the snapshot to the us-east-1 Region. Restore the database from the copied snapshot.
D) In the us-east-1 Region, choose to restore the latest automated backup of the database from the us-west-1 Region. Enable encryption using a KMS key in the us-east-1 Region.

7) A developer is adding Amazon ElastiCache for Memcached to a company's existing record storage application to reduce the load on the database and increase performance. The developer has decided to use lazy loading based on an analysis of common record handling patterns.

Which pseudocode example would correctly implement lazy loading?

A)
```
record_value = db.query("UPDATE Records SET Details = {1} WHERE ID == {0}",
record_key, record_value)
cache.set (record_key, record_value)
```
B)
```
record_value = cache.get(record_key)
if (record_value == NULL)
    record_value = db.query("SELECT Details FROM Records WHERE ID == {0}",
record_key)
    cache.set (record_key, record_value)
```
C)
```
record_value = cache.get (record_key)
db.query("UPDATE Records SET Details = {1} WHERE ID == {0}", record_key,
record_value)
```
D)
```
record_value = db.query("SELECT Details FROM Records WHERE ID == {0}",
record_key)
if (record_value != NULL)
    cache.set (record_key, record_value)
```

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved | aws.amazon.com

8) A developer wants to track the performance of an application that runs on a fleet of Amazon EC2 instances. The developer wants to view and track statistics across the fleet, such as the average and maximum request latency. The developer would like to be notified immediately if the average response time exceeds a threshold.

Which solution meets these requirements?

A) Configure a cron job on each instance to measure the response time and update a log file stored in an Amazon S3 bucket every minute. Use an Amazon S3 event notification to trigger an AWS Lambda function that reads the log file and writes new entries to an Amazon Elasticsearch Service (Amazon ES) cluster. Visualize the results in a Kibana dashboard. Configure Amazon ES to send an alert to an Amazon SNS topic when the response time exceeds a threshold.
B) Configure the application to write the response times to the system log. Install and configure the Amazon Inspector agent to continually read the logs and send the response times to Amazon EventBridge. View the metrics graphs in the EventBridge console. Configure an EventBridge custom rule to send an Amazon SNS notification when the average of the response time metric exceeds the threshold.
C) Configure the application to write the response times to a log file. Install and configure the Amazon CloudWatch agent on the instances to stream the application log to CloudWatch Logs. Create a metric filter of the response time from the log. View the metrics graphs in the CloudWatch console. Create a CloudWatch alarm to send an Amazon SNS notification when the average of the response time metric exceeds the threshold.
D) Install and configure the AWS Systems Manager Agent on the instances to monitor the response time and send it to Amazon CloudWatch as a custom metric. View the metrics graphs in Amazon QuickSight. Create a CloudWatch alarm to send an Amazon SNS notification when the average of the response time metric exceeds the threshold.

9) A developer is testing an application locally and has deployed it to AWS Lambda. To remain under the package size limit, the dependencies were not included in the deployment file. When testing the application remotely, the function does not execute because of missing dependencies.

Which approach would resolve the issue?

A) Use the Lambda console editor to update the code and include the missing dependencies.
B) Create an additional .zip file with the missing dependencies and include the file in the original Lambda deployment package.
C) Add references to the missing dependencies in the Lambda function's environment variables.
D) Attach a layer to the Lambda function that contains the missing dependencies.

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved | aws.amazon.com

10) A developer is building a web application that uses Amazon API Gateway. The developer wants to maintain different environments for development and production (dev and prod) workloads. The API will be backed by an AWS Lambda function with two aliases: one for dev and one for prod.

How can this be achieved with the LEAST amount of configuration?

A) Create a REST API for each environment and integrate the APIs with the corresponding dev and prod aliases of the Lambda function. Then deploy the two APIs to their respective stages and access them using the stage URLs.
B) Create one REST API and integrate it with the Lambda function using a stage variable in place of an alias. Then deploy the API to two different stages – dev and prod – and create a stage variable in each stage with different aliases as the values. Access the API using the different stage URLs.
C) Create one REST API and integrate it with the dev alias of the Lambda function, and deploy it to a dev environment. Configure a canary release deployment for prod where the canary will integrate with the Lambda prod alias.
D) Create one REST API and integrate it with the prod alias of the Lambda function and deploy it to a prod environment. Configure a canary release deployment for dev where the canary will integrate with the Lambda dev alias.

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved | aws.amazon.com

Answers

1) D – AWS Secrets Manager helps to protect the credentials needed to access databases, applications, services, and other IT resources. The service enables users to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to the Secrets Manager APIs, eliminating the need to hard code sensitive information in plaintext. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB.

2) A, B - AWS AppSync simplifies application development by letting users create a flexible API to securely access, manipulate, and combine data from one or more data sources. AWS AppSync is a managed service that uses GraphQL to make it easy for applications to get the exact data they need. AWS AppSync allows users to build scalable applications, including those requiring real-time updates, on a range of data sources, including Amazon DynamoDB. In Amazon API Gateway, users can create a WebSocket API as a stateful frontend for an AWS service (such as AWS Lambda or DynamoDB) or for an HTTP endpoint. The WebSocket API invokes the backend based on the content of the messages it receives from client applications. Unlike a REST API, which receives and responds to requests, a WebSocket API supports two-way communication between client applications and the backend.

3) A, E - Amazon Cognito adds user sign-up, sign-in, and access control to web and mobile applications quickly and easily. Users can also create an AWS Lambda function to make an API call to a custom analytics solution and then trigger that function with an Amazon Cognito post authentication trigger.

4) A, D - A resource policy can be used to grant API access to one AWS account to users in a different AWS account using Signature Version 4 (SigV4) protocols.

5) A - An AWS Lambda function's execution role grants it permission to access AWS services and resources. Users provide this role when a function is created, and Lambda assumes the role when a function is invoked.

6) A - If a user copies an encrypted snapshot, the copy of the snapshot must also be encrypted. If a user copies an encrypted snapshot across Regions, users cannot use the same AWS KMS encryption key for the copy as used for the source snapshot, because KMS keys are Region-specific. Instead, users must specify a KMS key that is valid in the destination Region.

7) B - Lazy loading is a concept where the loading of a record is delayed until it is needed. Lazy loading first checks the cache. If a record is not present, lazy loading retrieves the record from the database, and then stores the record in the cache.

8) C – The Amazon CloudWatch Agent can be configured to stream logs and metrics to CloudWatch. Metric filters can be created from logs stored in CloudWatch Logs.

9) D - Users can configure an AWS Lambda function to pull in additional code and content in the form of layers. A layer is a .zip archive that contains libraries, a custom runtime, or other dependencies. With layers, users can use libraries in a function without needing to include the libraries in a deployment package.

10) B - With deployment stages in Amazon API Gateway, users can manage multiple release stages for each API, such as alpha, beta, and production. Using stage variables that can be configured, an API deployment stage can interact with different backend endpoints. Users can use API Gateway stage variables to reference a single AWS Lambda function with multiple versions and aliases.

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved | aws.amazon.com