AWS DVA-C02

AWS Developer Associate Certification Questions & Answers

Get Instant Access to Vital Exam Acing Materials | Study Guide | Sample Questions | Practice Test

DVA-C02

AWS Certified Developer - Associate

65 Questions Exam – 720 / 1000 Cut Score – Duration of 130 minutes

Table of Contents:

Discover More about the DVA-C02 Certification

Are you interested in passing the AWS DVA-C02 exam? First discover, who benefits from the DVA-C02 certification. The DVA-C02 is suitable for a candidate if he wants to learn about Associate. Passing the DVA-C02 exam earns you the AWS Certified Developer - Associate title.

While preparing for the DVA-C02 exam, many candidates struggle to get the necessary materials. But do not worry; your struggling days are over. The DVA-C02 PDF contains some of the most valuable preparation tips and the details and instant access to useful DVA-C02 study materials **just at one click.**

AWS DVA-C02 Developer Associate Certification Details:

Exam Name	AWS Developer Associate (AWS-CDA)
Exam Code	DVA-C02
Exam Price	$150 USD
Duration	130 minutes
Number of Questions	65
Passing Score	720 / 1000
Recommended Training / Books	**Developing on AWS**
Schedule Exam	**AWS Certification**
Sample Questions	**AWS DVA-C02 Sample Questions**
Recommended Practice	**AWS Certified Developer - Associate Practice Test**

DVA-C02 Syllabus:

Section	Objectives
Development with AWS Services - 32%	
Develop code for applications hosted on AWS.	Knowledge of: • Architectural patterns (for example, event-driven,

Section	Objectives
	microservices, monolithic, choreography, orchestration, fanout) • Idempotency • Differences between stateful and stateless concepts • Differences between tightly coupled and loosely coupled components • Fault-tolerant design patterns (for example, retries with exponential backoff and jitter, dead-letter queues) • Differences between synchronous and asynchronous patterns Skills in: • Creating fault-tolerant and resilient applications in a programming language (for example, Java, C#, Python, JavaScript, TypeScript, Go) • Creating, extending, and maintaining APIs (for example, response/request transformations, enforcing validation rules, overriding status codes) • Writing and running unit tests in development environments (for example, using AWS Serverless Application Model [AWS SAM]) • Writing code to use messaging services • Writing code that interacts with AWS services by using APIs and AWS SDKs • Handling data streaming by using AWS services
Develop code for AWS Lambda.	Knowledge of: • Event source mapping • Stateless applications • Unit testing • Event-driven architecture • Scalability • The access of private resources in VPCs from Lambda code Skills in: • Configuring Lambda functions by defining environment variables and parameters (for example, memory, concurrency, timeout, runtime, handler, layers,

Section	Objectives
	extensions, triggers, destinations) • Handling the event lifecycle and errors by using code (for example, Lambda Destinations, dead-letter queues) • Writing and running test code by using AWS services and tools • Integrating Lambda functions with AWS services • Tuning Lambda functions for optimal performance
Use data stores in application development.	Knowledge of: • Relational and non-relational databases • Create, read, update, and delete (CRUD) operations • High-cardinality partition keys for balanced partition access • Cloud storage options (for example, file, object, databases) • Database consistency models (for example, strongly consistent, eventually consistent) • Differences between query and scan operations • Amazon DynamoDB keys and indexing • Caching strategies (for example, write-through, read-through, lazy loading, TTL) • Amazon S3 tiers and lifecycle management • Differences between ephemeral and persistent data storage patterns Skills in: • Serializing and deserializing data to provide persistence to a data store • Using, managing, and maintaining data stores • Managing data lifecycles • Using data caching services
Security - 26%	
Implement authentication and/or authorization for applications and AWS services.	Knowledge of: • Identity federation (for example, Security Assertion Markup Language [SAML], OpenID Connect [OIDC], Amazon Cognito)

Section	Objectives
	• Bearer tokens (for example, JSON Web Token [JWT], OAuth, AWS Security Token Service [AWS STS]) • The comparison of user pools and identity pools in Amazon Cognito • Resource-based policies, service policies, and principal policies • Role-based access control (RBAC) • Application authorization that uses ACLs • The principle of least privilege • Differences between AWS managed policies and customer-managed policies • Identity and access management (IAM) Skills in: • Using an identity provider to implement federated access (for example, Amazon Cognito, AWS Identity and Access Management [IAM]) • Securing applications by using bearer tokens • Configuring programmatic access to AWS • Making authenticated calls to AWS services • Assuming an IAM role • Defining permissions for principals
Implement encryption by using AWS services.	Knowledge of: • Encryption at rest and in transit • Certificate management (for example, AWS Private Certificate Authority) • Key protection (for example, key rotation) • Differences between client-side encryption and server-side encryption • Differences between AWS managed and customer managed AWS Key Management Service (AWS KMS) keys Skills in: • Using encryption keys to encrypt or decrypt data • Generating certificates and SSH keys for development purposes

Section	Objectives
	• Using encryption across account boundaries • Enabling and disabling key rotation
Manage sensitive data in application code.	Knowledge of: • Data classification (for example, personally identifiable information [PII], protected health information [PHI]) • Environment variables • Secrets management (for example, AWS Secrets Manager, AWS Systems Manager Parameter Store) • Secure credential handling Skills in: • Encrypting environment variables that contain sensitive data • Using secret management services to secure sensitive data • Sanitizing sensitive data
Deployment - 24%	
Prepare application artifacts to be deployed to AWS.	Knowledge of: • Ways to access application configuration data (for example, AWS AppConfig, Secrets Manager, Parameter Store) • Lambda deployment packaging, layers, and configuration options • Git-based version control tools (for example, Git, AWS CodeCommit) • Container images Skills in: • Managing the dependencies of the code module (for example, environment variables, configuration files, container images) within the package • Organizing files and a directory structure for application deployment • Using code repositories in deployment environments • Applying application requirements for resources (for example, memory, cores)

Section	Objectives
Test applications in development environments.	Knowledge of: • Features in AWS services that perform application deployment • Integration testing that uses mock endpoints • Lambda versions and aliases Skills in: • Testing deployed code by using AWS services and tools • Performing mock integration for APIs and resolving integration dependencies • Testing applications by using development endpoints (for example, configuring stages in Amazon API Gateway) • Deploying application stack updates to existing environments (for example, deploying an AWS SAM template to a different staging environment)
Automate deployment testing.	Knowledge of: • API Gateway stages • Branches and actions in the continuous integration and continuous delivery (CI/CD) workflow • Automated software testing (for example, unit testing, mock testing) Skills in: • Creating application test events (for example, JSON payloads for testing Lambda, API Gateway, AWS SAM resources) • Deploying API resources to various environments • Creating application environments that use approved versions for integration testing (for example, Lambda aliases, container image tags, AWS Amplify branches, AWS Copilot environments) • Implementing and deploying infrastructure as code (IaC) templates (for example, AWS SAM templates, AWS CloudFormation templates) • Managing environments in individual AWS services (for example, differentiating between development, test, and production in API Gateway)

Section	Objectives
Deploy code by using AWS CI/CD services.	Knowledge of: • Git-based version control tools (for example, Git, AWS CodeCommit) • Manual and automated approvals in AWS CodePipeline • Access application configurations from AWS AppConfig and Secrets Manager • CI/CD workflows that use AWS services • Application deployment that uses AWS services and tools (for example, CloudFormation, AWS Cloud Development Kit [AWS CDK], AWS SAM, AWS CodeArtifact, AWS Copilot, Amplify, Lambda) • Lambda deployment packaging options • API Gateway stages and custom domains • Deployment strategies (for example, canary, blue/green, rolling) Skills in: • Updating existing IaC templates (for example, AWS SAM templates, CloudFormation templates) • Managing application environments by using AWS services • Deploying an application version by using deployment strategies • Committing code to a repository to invoke build, test, and deployment actions • Using orchestrated workflows to deploy code to different environments • Performing application rollbacks by using existing deployment strategies • Using labels and branches for version and release management • Using existing runtime configurations to create dynamic deployments (for example, using staging variables from API Gateway in Lambda functions)
Troubleshooting and Optimization - 18%	
Assist in a root cause analysis.	Knowledge of:

Section	Objectives
	• Logging and monitoring systems • Languages for log queries (for example, Amazon CloudWatch Logs Insights) • Data visualizations • Code analysis tools • Common HTTP error codes • Common exceptions generated by SDKs • Service maps in AWS X-Ray Skills in: • Debugging code to identify defects • Interpreting application metrics, logs, and traces • Querying logs to find relevant data • Implementing custom metrics (for example, CloudWatch embedded metric format [EMF]) • Reviewing application health by using dashboards and insights • Troubleshooting deployment failures by using service output logs
Instrument code for observability.	Knowledge of: • Distributed tracing • Differences between logging, monitoring, and observability • Structured logging • Application metrics (for example, custom, embedded, built-in) Skills in: • Implementing an effective logging strategy to record application behavior and state • Implementing code that emits custom metrics • Adding annotations for tracing services • Implementing notification alerts for specific actions (for example, notifications about quota limits or deployment completions) • Implementing tracing by using AWS services and tools

Section	Objectives
Optimize applications by using AWS services and features.	Knowledge of: • Caching • Concurrency • Messaging services (for example, Amazon Simple Queue Service [Amazon SQS], Amazon Simple Notification Service [Amazon SNS]) Skills in: • Profiling application performance • Determining minimum memory and compute power for an application • Using subscription filter policies to optimize messaging • Caching content based on request headers

Broaden Your Knowledge with AWS DVA-C02 Sample Questions:

Question: 1

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

What should the developer do to ensure that the Lambda function has the correct permissions?

a) Create a Lambda execution role by using AWS Identity and Access Management (IAM). Attach the IAM policy to the role. Assign the Lambda execution role to the Lambda function.
b) Create a Lambda execution user by using AWS Identity and Access Management (IAM). Attach the IAM policy to the user. Assign the Lambda execution user to the Lambda function.
c) Create a Lambda execution role by using AWS Identity and Access Management (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 by using AWS Identity and Access Management (IAM). Attach the IAM policy to the user. Store the IAM user credentials as environment variables in the Lambda function.

Answer: a

Question: 2

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

How can the developer maintain these environments with the LEAST amount of configuration?

a) Create a REST API for each environment. Integrate the APIs with the corresponding dev and prod aliases of the Lambda function. Deploy the APIs to their respective stages. Access the APIs by using the stage URLs.
b) Create one REST API. Integrate the API with the Lambda function by using a stage variable in place of an alias. Deploy the API to two different stages: dev and prod. Create a stage variable in each stage with different aliases as the values. Access the API by using the different stage URLs.
c) Create one REST API. Integrate the API with the dev alias of the Lambda function. Deploy the API to the dev environment. Configure a canary release deployment for the prod environment where the canary will integrate with the Lambda prod alias.
d) Create one REST API. Integrate the API with the prod alias of the Lambda function. Deploy the API to the prod environment. Configure a canary release deployment for the dev environment where the canary will integrate with the Lambda dev alias.

Answer: b

Question: 3

A developer is creating a web application that must give users the ability to post comments and receive feedback in near real time. Which solutions will 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. Use an Amazon DynamoDB table as the data store.
c) Create an AWS Elastic Beanstalk application that is 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) Establish WebSocket connections to Amazon CloudFront. Use an AWS Lambda function as the CloudFront distribution's origin. Use an Amazon Aurora DB cluster as the data store.

Answer: a, b

Question: 4

A company is migrating a legacy application to Amazon EC2 instances. The application uses a user name and password that are stored in the source code to connect to a MySQL database.

The company will migrate the database to an Amazon RDS for MySQL DB instance. As part of the migration, the company needs to implement a secure way to store and automatically rotate the database credentials.

Which solution will meet 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.

Answer: d

Question: 5

A developer is adding Amazon ElastiCache for Memcached to a company's existing record storage application. The developer has decided to use lazy loading based on an analysis of common record handling patterns. Which pseudocode example will 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)

Answer: b

Question: 6

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, such as the average request latency and the maximum request latency, across the fleet. The developer wants to receive immediate notification if the average response time exceeds a threshold.

Which solution will meet these requirements?

a) Configure a cron job on each EC2 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 invoke an AWS Lambda function that reads the log file and writes new entries to an Amazon OpenSearch Service cluster. Visualize the results in OpenSearch Dashboards. Configure OpenSearch Service to send an alert to an Amazon Simple Notification Service (Amazon SNS) topic when the response time exceeds the threshold.

b) Configure the application to write the response times to the system log. Install and configure the Amazon Inspector agent on the EC2 instances to continually read the logs and send the response times to Amazon EventBridge (Amazon CloudWatch Events). View the metrics graphs in the EventBridge (CloudWatch Events) console. Configure an EventBridge (CloudWatch Events) custom rule to send an Amazon Simple Notification Service (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 EC2 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 Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold.

d) Install and configure AWS Systems Manager Agent (SSM Agent) on the EC2 instances to monitor the response time and send the response time to Amazon CloudWatch as a custom metric. View the metrics graphs in Amazon QuickSight. Create a CloudWatch alarm to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold.

Answer: c

Question: 7

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

Which combination of Steps should the developer take to meet these requirements?

(Select TWO.)

a) Create an IAM permission policy. Attach the policy to each IAM user. Set the method authorization type for the APIs to AWS_IAM. Use Signature Version 4 to sign the API requests.
b) Create an Amazon Cognito user pool. Add each IAM user to the user pool. Set the method authorization type for the APIs to COGNITO_USER_POOLS. Authenticate by using the IAM credentials in Amazon Cognito. Add the ID token to the request headers.
c) Create an Amazon Cognito identity pool. Add each IAM user to the identity pool. Set the method authorization type for the APIs to COGNITO_USER_POOLS. Authenticate by using the IAM credentials in Amazon Cognito. Add the access token to the request headers.
d) Create a resource policy for the APIs to allow access for each IAM user only.
e) Create an Amazon Cognito authorizer for the APIs to allow access for each IAM user only. Set the method authorization type for the APIs to COGNITO_USER_POOLS.

Answer: a, d

Question: 8

A developer is testing an application locally and has deployed the application to an AWS Lambda function. To avoid exceeding the deployment package size quota, the developer did not include the dependencies in the deployment file.

When the developer tests the application remotely, the Lambda function does not run because of missing dependencies. Which solution will resolve this issue?

a) Use the Lambda console editor to update the code and include the missing dependencies.
b) Create an additional .zip file that contains the missing dependencies. Include the .zip file in the original Lambda deployment package.
c) Add references to the missing dependencies in the Lambda function's environment variables.
d) Create a layer that contains the missing dependencies. Attach the layer to the Lambda function.

Answer: d

Question: 9

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

Which combination of actions should the developer perform to meet these requirements?

(Select TWO.)

a) Use Amazon Cognito to provide the sign-up and sign-in functionality.
b) Use AWS Identity and Access Management (IAM) to provide the sign-up and sign-in functionality.
c) Configure an AWS Config rule to make the API call when a user is authenticated.
d) Invoke an Amazon API Gateway method to make the API call when a user is authenticated.
e) Invoke an AWS Lambda function to make the API call when a user is authenticated.

Answer: a, e

Question: 10

A developer is working on an application that stores highly confidential data in a database. The developer must use AWS Key Management Service (AWS KMS) with envelope encryption to protect the data.

How should the developer configure the data encryption to meet these requirements?

a) Encrypt the data by using a KMS key. Store the encrypted data in the database.
b) Encrypt the data by using a generated data key. Store the encrypted data in the database.
c) Encrypt the data by using a generated data key. Store the encrypted data and the data key ID in the database.
d) Encrypt the data by using a generated data key. Store the encrypted data and the encrypted data key in the database.

Answer: d

Avail the Study Guide to Pass AWS DVA-C02 Developer Associate Exam:

- Find out about the DVA-C02 syllabus topics. Visiting the official site offers an idea about the exam structure and other important study resources. Going through the syllabus topics help to plan the exam in an organized manner.
- Once you are done exploring the **DVA-C02 syllabus**, it is time to plan for studying and covering the syllabus topics from the core. Chalk out the best plan for yourself to cover each part of the syllabus in a hassle-free manner.
- A study schedule helps you to stay calm throughout your exam preparation. It should contain your materials and thoughts like study hours, number of topics for daily studying mentioned on it. The best bet to clear the exam is to follow your schedule rigorously.
- The candidate should not miss out on the scope to learn from the DVA-C02 training. Joining the AWS provided training for DVA-C02 exam helps a candidate to strengthen his practical knowledge base from the certification.
- Learning about the probable questions and gaining knowledge regarding the exam structure helps a lot. Go through the **DVA-C02 sample questions** and boost your knowledge
- Make yourself a pro through online practicing the syllabus topics. DVA-C02 practice tests would guide you on your strengths and weaknesses regarding the syllabus topics. Through rigorous practicing, you can improve the weaker sections too. Learn well about time management during exam and become confident gradually with practice tests.

Career Benefits:

- Passing the DVA-C02 exam, helps a candidate to prosper highly in his career. Having the certification on the resume adds to the candidate's benefit and helps to get the best opportunities.

Here Is the Trusted Practice Test for the DVA-C02 Certification

VMExam.Com is here with all the necessary details regarding the DVA-C02 exam. We provide authentic practice tests for the DVA-C02 exam. What do you gain from these practice tests? You get to experience the real exam-like questions made by industry experts and get a scope to improve your performance in the actual exam. Rely on VMExam.Com for rigorous, unlimited two-month attempts on the **DVA-C02 practice tests**, and gradually build your confidence. Rigorous practice made many aspirants successful and made their journey easy towards grabbing the AWS Certified Developer - Associate.

Start Online practice of DVA-C02 Exam by visiting URL
https://www.vmexam.com/aws/dva-c02-aws-developer-associate