

AWS Certified Developer - Associate

Domínios, pesos oficiales y habilidades clave del temario de DVA-C02, para repasar antes del examen.

Nivel: Intermediate (Associate) Duración: 130 min Nota de corte: 720/1000 Fuente: guía de estudio oficial de Microsoft Learn

1. Development with AWS Services

32–32%

Develop code for applications hosted on AWS

- Un fanout de SNS a varias colas SQS entrega una copia de cada mensaje a cada consumidor y la conserva hasta que la procesa. En las sagas, la coreografía encadena servicios mediante eventos (por ejemplo, con EventBridge) sin coordinador central, y la orquestación usa un coordinador como Step Functions que ejecuta los pasos y las compensaciones. El patrón singleton migra un monolito por partes con un proxy como API Gateway.
- Un componente sin estado no guarda en memoria ni en disco local nada que necesite una solicitud posterior, así que cualquier instancia puede atenderla. Las sesiones web van a un almacén central como ElastiCache; las sesiones persistentes del ALB no eliminan el estado. En Lambda los entornos se reutilizan (conexiones fuera del controlador, cache system), pero este estado no va a un solo lugar. Un componente obliga a cambiar o hace fallar a los que dependen de él. El acoplamiento débil añade una capa intermedia (SQS, Kinesis, Step Functions, EventBridge o SNS) y hace que cada componente solo conozca una interfaz publicada y versionada. Compartir una base de datos entre servicios y las llamadas directas sin procesamiento asincrónico son antipáternes.
- En el servicio de invocación de funciones, Lambda se invoca de forma síncrona con RequestResponse y asíncrona con Event, y la asíncrona puede enviar los fallos a destinos como colas estándar de SQS. API Gateway invoca Lambda de forma asíncrona con el encabezado X-Amz-Invocation-Type. EventBridge transmite sin limitación con retroceso exponencial y jitter; errores como ValidationException no se reintentan. El modo standard es el recomendado y el adaptive añade un limitador de ritmo en el cliente. La configuración en código prevalece sobre las variables de entorno y el archivo compartido. Como puede haber duplicados, las escrituras deben ser idempotentes.

Develop code for AWS Lambda

- Para que Lambda llegue a recursos privados de una VPC (RDS, ElastiCache), la función se conecta a las subredes privadas con un grupo de seguridad permitido, y su rol necesita permisos de interfaces de red de EC2, como los de AWSLambdaVPCAccessExecutionRole. Una función en una VPC solo ve esa VPC: para salir a Internet necesita una puerta de enlace NAT.
- Una política de permisos IAM, Policy, de la configuración de cada versión y update-function-configuration sustituye toda la estructura Variables. Las capas comparten dependencias entre funciones .zip y se actualizan aparte del código. La simultaneidad reservada fija un mínimo y un máximo para la función; la aprovisionada mantiene entornos.
- En invocaciones asíncronas, Lambda ignora los errores y puede enviar un registro con la solicitud y la respuesta a un destino en caso de error; la cola de mensajes fallidos solo recibe el evento. Con SQS, ReportBatchItemFailures reintenta solo los mensajes fallidos. Con Kinesis conviene limitar MaximumRetryAttempts y añadir un destino para que un registro no se pierda por un fallo de fragmento.
- AWS SAM CLI permite generar partes de la estructura real (sam local generate-event) e invocar la función en un contenedor local (sam local invoke), aunque estas pruebas no validan permisos entre recursos. Para eso hay que probar en la nube, por ejemplo con sam remote invoke. La consola admite eventos de prueba privados y servicios como S3 y SNS invocan Lambda de forma asíncrona; API Gateway lo hace de forma síncrona; SQS, Kinesis y DynamoDB Streams usan una asignación de origen de eventos que sondea y entrega lotes. Los servicios que invocan una función necesitan permiso en su política basada en recursos. Si una función escribe en el bucket que la activa, conviene usar otro bucket o un prefijo para evitar bucles.

Use data stores in application development

- La clave de partición determina en qué partición se guarda cada elemento; cuantos más valores distintos se usen, mejor se reparte el tráfico. Un ID de usuario es buena clave; un estado con pocos valores o una fecha redondeada crean particiones calientes. Para repartir escrituras se puede añadir un sufijo aleatorio, que obliga a consultar todos los datos.
- En DynamoDB las lecturas son de coherencia final por defecto, con ConsistentRead en true, GetItem, Query y Scan devuelven los datos más recientes, pero solo en tablas e índices secundarios locales. Los índices globales y los flujos son siempre de coherencia final. Las tablas globales usan MREC (replicación asíncrona) o MRSC (síncrona), donde se lee solo los elementos de un valor de clave de partición, con una condición opcional sobre la clave de ordenación, y devuelve los resultados ordenados. Scan recorre toda la tabla o el índice y es menos eficiente a medida que crece. En ambos casos el filtro se aplica después de leer, así que no reduce la capacidad consumida.
- Las claves principales, simple (partición) o compuesta (partición y ordenación). Un índice secundario local mantiene la clave de partición con otra clave de ordenación y se crea con la tabla; uno global puede tener claves distintas y añadirse después con UpdateTable. Las proyecciones KEYS_ONLY, INCLUDE o ALL fijan qué atributos se copian, y el resto de atributos de clave faltan en muchos elementos es disperso.
- El API de flujo de datos de DynamoDB usa DynamoDB JSON en dispositivos de tipo como S o N y los números enviados como cadenas. En JavaScript v3, DynamoDBDocumentClient evita serializar a mano y util-dynamodb ofrece marshall y unmarshall. En Java 2.x, el cliente mejorado asigna clases anotadas con @DynamoDbBean a tablas mediante TableSchema.fromBean.

Implement authentication and/or authorization for applications and AWS services

- Los grupos de identidades de Amazon Cognito intercambian tokens de proveedores sociales, SAML u OIDC (o de un grupo de usuarios) por credenciales temporales de AWS ligadas a roles de IAM. Los grupos de usuarios federan IdP SAML u OIDC y emiten sus propios JWT para la aplicación. Para cargas como GitHub Actions, se crea un proveedor OIDC en IAM y un rol con sts:AssumeRoleWithWebIdentity limitado por condiciones como sub.
- Los grupos de usuarios de Cognito emiten tokens de ID (identidad), de acceso (ámbitos) y de actualización. En una API REST, un autorizador COGNITO_USER_POOLS valida el token de la cabecera Authorization; en una API HTTP, el autorizador JWT comprueba firma, iss, aud o client_id, caducidad y los ámbitos de la ruta. Fuera de API Gateway, valida la firma con el JWKS, por ejemplo con aws-jwt-verify.
- Para acceso programático, AWS recomienda credenciales temporales: perfiles de IAM Identity Center en la CLI (aws configure sso y aws sso login), roles de perfil de instancia o de tarea en cómputo de AWS e IAM Roles Anywhere fuera de AWS. La CLI y los SDK resuelven credenciales en orden: opciones de línea de comandos, variables de entorno, archivos compartidos y, al final, credenciales de contenedor o de instancia.
- Las peticiones a las APIs de AWS se autentican con Signature Version 4: el SDK o la CLI firman con las credenciales y AWS recalcula la firma, limitada a fecha, región y servicio. Las URL de función de Lambda con AWS_IAM exigen peticiones firmadas con SigV4. Las URL prefiradas de S3 usan las credenciales de quien las creó y caducan con ellas si son temporales; un reloj desajustado provoca errores de firma caducada.
- Asumir un rol devuelve credenciales temporales de STS. El rol tiene una política de confianza (quién puede asumirlo) y una de permisos (qué puede hacer); entre cuentas, quien llama necesita además permiso sts:AssumeRole sobre el ARN. Las políticas de sesión solo reducen permisos, sts:ExternalId evita el suplente confuso con terceros y la CLI asume roles con role_arn más source_profile o credential_source.

Implement encryption by using AWS services

- En reposo: S3 cifra los objetos nuevos con SSE-S3 por defecto y admite SSE-KMS; DynamoDB cifra siempre las tablas y permite cambiar entre clave propiedad de AWS, administrada por AWS o por el cliente; RDS solo se cifra al crear la instancia, o copiando una instantánea con cifrado y restaurándola. En tránsito: una política de bucket que deniega peticiones con aws:SecureTransport igual a false obliga a usar TLS.
- AWS Certificate Manager emite certificados públicos y los renueva automáticamente si se validan por DNS y el registro CNAME se mantiene; con validación por correo la renovación exige acción del propietario y los certificados importados se renuevan reimportándolos con el mismo ARN. AWS Private CA crea jerarquías privadas de CA raíz y subordinadas que emiten certificados X.509 para TLS interno, con OCSP.
- Con cifrado del lado del servidor, el servicio cifra los datos al escribirlos y los descifra de forma transparente para cualquier petición autorizada (SSE-S3, SSE-KMS, DSSE-KMS o SSE-C en S3, cifrado en reposo en DynamoDB). Con cifrado del lado del cliente, la aplicación cifra antes de enviar, con el Amazon S3 Encryption Client o el AWS Database Encryption SDK, y el servicio solo almacena texto cifrado.
- El cifrado de sobre de AWS KMS usa GenerateDataKey: la aplicación cifra localmente con la clave de datos sin cifrar, la borra y guarda la copia cifrada junto a los datos; GenerateDataKeyWithoutPlaintext devuelve solo la copia cifrada. Para descifrar se llama a Decrypt, donde KeyId es opcional con claves simétricas y el contexto de cifrado debe coincidir exactamente, porque va unido al texto cifrado como AAD.
- Para desarrollo, Amazon EC2 crea pares de claves RSA o ED25519 con create-key-pair y entrega la clave privada una sola vez sin guardar copia; también puedes generarlas con ssh-keygen e importar solo la pública con import-key-pair. EC2 Instance Connect envía claves públicas temporales controladas por IAM. ACM puede importar certificados autofirmados si la clave privada está sin cifrar y en PEM.

Manage sensitive data in application code

- Clasificar los datos (PII, PHI, financieros) define qué controles necesita cada uno: acceso, cifrado, retención y auditoría. AWS recomienda etiquetar los recursos con su clasificación, vigilarlos con AWS Config y fijar las claves permitidas con políticas de etiquetas. Amazon Macie localiza datos confidenciales en S3 y Comprehend Medical DetectPHI detecta PHI en texto clínico.
- Lambda cifra siempre en reposo las variables de entorno con KMS: por defecto con una clave administrada por AWS, o con una administrada por el cliente (KMSKeyArn) si se quiere controlar la rotación. Las ayudas de cifrado en tránsito cifran el valor en el cliente, así que el código debe llamar a Decrypt y el rol necesita kms:Decrypt. Denegar kms:Decrypt impide ver las variables sin impedir administrar la función.
- Secrets Manager es el servicio indicado para credenciales, claves de API y tokens: ofrece rotación automática y versiones con etiquetas como AWSCURRENT; Parameter Store SecureString sirve para configuración cifrada sin rotación. En Lambda, la extensión Parameters and Secrets almacena secretos en caché vía HTTP local. En ECS, los secretos inyectados se leen al arrancar la tarea y hay que relanzarla tras rotarlos.
- Las políticas de protección de datos de CloudWatch Logs detectan y enmascaran al ingerir datos como tarjetas, PII, PHI o credenciales; se aplican a un grupo de registros o a toda la cuenta y solo quien tiene logs:Unmask ve los valores originales. Los eventos anteriores a la política no se enmascaran. En Step Functions, includeExecutionData en false excluye la entrada y salida de los registros.
- El enmascaramiento en la aplicación puede ser irreversible o reversible. La utilidad de Powertools for AWS Lambda ofrece erase, que sustituye campos concretos por un marcador, y encrypt/decrypt con KMS y contexto de cifrado, que debe coincidir al descifrar. Amazon Comprehend localiza PII en tiempo real (DetectPiiEntities) y genera copias redactadas con un trabajo asíncrono en modo ONLY_REDACTION.

3. Deployment

24-24%

Prepare application artifacts to be deployed to AWS

- Las capas de Lambda empaquetan bibliotecas, entornos de ejecución personalizados o archivos de configuración en un .zip que varias funciones comparten y que se actualiza sin tocar el código; solo sirven para funciones .zip.
- En las funciones con imagen de contenedor, las dependencias van dentro de la imagen. Las dependencias nativas deben compilarse para Linux, por ejemplo con `sudo --use-container`.
- En Lambda, la memoria es el único ajuste de cómputo: la CPU se asigna en proporción a ella, y el espacio de /tmp se configura aparte con el almacenamiento efímero. En ECS, la definición de tareas fija cpu y memory a nivel de tarea o de contenedor; memory es un límite estricto que elimina el contenedor si lo supera y memoryReservation es un límite flexible que se usa para la colocación.
- AWS AppConfig separa la configuración del código: una aplicación tiene entornos (por ejemplo Beta y Production), es un límite flexible que se usa para la colocación.

Si una alarma de CloudWatch del entorno salta, la configuración se revierte. En Lambda, la extensión AppConfig

Test applications in development environments

- Para probar código ya implementado: `aws remote invoke` invoca funciones de Lambda, envía mensajes a SQS o registros a Kinesis e inicia máquinas de estados en la nube; la API `TestState` de Step Functions prueba un estado sin crear ni modificar la máquina, con niveles INFO, DEBUG y TRACE; y la pestaña Test de API Gateway (o `test-invoke-method`) ejecuta llamadas reales con registros simulados.
- o dependencias externas: API Gateway ofrece integraciones simuladas sin backend y la API `TestState` de Step Functions permite simular resultados o errores de una integración.
- En API Gateway cada etapa es una referencia con nombre a una implementación y forma parte de la URL de invocación, por lo que una etapa dev sirve como endpoint de pruebas. `TestInvokeMethod` y la pestaña Test de la consola ejecutan un método sin implementarlo, pero la llamada es real. En las API HTTP, la etapa `$default` se sirve en la URL base y las implementaciones automáticas publican cada cambio.
- El archivo `aws-samconfig.toml` admite varios entornos con nombre, y `aws deploy --config-env` elige cuál usar (nombre de pila, región, `parameter_overrides`). Los valores de la línea de comandos tienen prioridad sobre el archivo y este sobre los valores predeterminados de la plantilla. `aws sync` acelera las pruebas pero se recomienda solo para desarrollo, staging y producción se actualizan con `aws deploy` o una canalización.
- Para probar flujos basados en eventos hay que comprobar el efecto en los servicios reales. `aws remote invoke` envía mensajes a colas de SQS, flujos de Kinesis o máquinas de estados de una pila desplegada. `TestEventPattern` indica si un evento coincide con el patrón de una regla de EventBridge sin entregarlo, y los archivos de EventBridge

Automate deployment events

- Un evento de prueba es la entrada JSON de una función. `aws local generate-event` crea cargas con la misma estructura que los eventos reales de cada servicio y admite opciones para cambiar sus valores. Los eventos de prueba compartibles de Lambda se guardan en el registro de esquemas de EventBridge, requieren permisos de
- Para llevar una API REST a otro entorno se crea una implementación y se asocia a una etapa, cualquier cambio en ese registro y se gestionan desde la CLI con `aws remote test-event`.
- En métodos, integraciones, autorizadores o políticas de recursos exige volver a implementar, mientras que la configuración de etapa o las asignaciones de ruta base no. Las versiones canary prueban una implementación con parte del tráfico de una etapa, y la exportación OpenAPI con extensiones replica la API.
- Un entorno de integración debe apuntar a versiones aprobadas y estables. En Lambda se usa un alias que apunta a una versión publicada y se mueve tras cada aprobación, porque el ARN no calificado invoca \$LATEST. En Amazon ECR, la inmutabilidad de etiquetas impide reutilizar una etiqueta aprobada. En Amplify Hosting cada rama conectada obtiene su propia implementación y URL.
- Una plantilla de AWS SAM es una plantilla de CloudFormation con la declaración obligatoria `Transform: AWS::Serverless-2016-10-31`. Los artefactos locales se suben a S3 con `aws cloudformation package` (o `aws deploy`) antes de desplegar. Al crear la pila hay que reconocer las capacidades necesarias: `CAPABILITY_IAM` o `CAPABILITY_AUTOPILOT`.
- En API Gateway, el `aws-samconfig.toml` define el uso de recursos de API, `CAPABILITY_IAM` o `CAPABILITY_AUTOPILOT` define limitaciones, registros, seguimiento de datos y variables de etapa. Las variables de etapa permiten que dev y prod usen backends, alias de Lambda o tablas distintos con la misma API, pero no deben contener datos sensibles y, si eligen una función

Deploy and by using AWS Continuous Integration and Continuous Delivery (CI/CD) services

- Lambda admite dos tipos de paquete: archivo .zip e imagen de contenedor almacenada en Amazon ECR, y el tipo no se puede cambiar en una función existente. Las capas son archivos .zip con dependencias compartidas que se extraen en /opt y solo sirven para funciones .zip. Las imágenes deben implementar la API de tiempo de ejecución; con una imagen base ajena a AWS hay que añadir el cliente de la interfaz de tiempo de ejecución.
- Un dominio personalizado sustituye ese nombre de host: requiere un certificado de ACM (en la misma región para dominios regionales), un registro DNS hacia el dominio de API Gateway y asignaciones de API a etapas, que eliminan la ruta base. Se puede desactivar el endpoint predeterminado para forzar el uso del dominio, modificarlo o sustituirlo. Cada propiedad tiene un comportamiento de actualización; si provoca Replacement, UpdateReplacePolicy decide si el recurso antiguo se conserva, se guarda en instantánea o se elimina. En
- La configuración de pila y de despliegue se organiza en una cadena de `ParameterStore` (dev/dep/prod) gradual con `GetParametersByPath`. CodePipeline separa los entornos en etapas y una acción de aprobación manual detiene la ejecución hasta que alguien con permisos la aprueba. En Elastic Beanstalk, un despliegue azul/verde prueba la versión en otro entorno e intercambia los CNAME con `Swap environment URLs`.
- Elastic Beanstalk despliega versiones con las políticas All at once, Rolling, Rolling with additional batch (mantiene la capacidad), Immutable y Traffic Splitting (canary con ALB). Para cambiar a una versión de plataforma incompatible se usa blue/green con intercambio de CNAME. En Lambda, un alias ponderado reparte el tráfico entre dos versiones publicadas.

4. Troubleshooting and Optimization

18–18%

Assist in a root cause analysis

- Muchos defectos de Lambda salen del código: en Node.js, no usar `await` hace que el handler termine antes de que acaben las llamadas; las claves de objeto de los eventos de S3 llegan codificadas como URL y hay que decodificarlas; y escribir en el mismo bucket o tabla que dispara la función crea bucles infinitos. La línea `START` del log indica qué versión se invocó.
- En Lambda, `Errors` cuenta errores de función (incluidos tiempos de espera), `Throttles` cuenta peticiones rechazadas que no aparecen en `Invocations` ni `Errors`, e `IteratorAge` mide el retraso de los registros de flujos. En X-Ray, `Error` son errores 4xx, `Fault` errores 5xx y `Throttle` respuestas 429; los servicios sin segmentos propios aparecen como segmentos inferidos.
- `CloudWatch Logs Insights` consulta logs ya almacenados con comandos encadenados con `|`: `fields`, `filter`, `parse` (glob o regex), `stats` con `bin()`, `sort`, `limit` y `dedup`. Los filtros de métricas, en cambio, no son retroactivos: solo generan datos desde que se crean. Para limitar el coste, consulta solo los grupos de logs necesarios y el intervalo de tiempo más estrecho.
- El formato de métricas incrustadas (EMF) crea métricas personalizadas de forma asíncrona a partir de logs JSON con un nodo `_aws.CloudWatchMetrics` (`Namespace`, `Dimensions`, `Metrics`). Solo requiere `logs:PutLogEvents`, no `cloudwatch:PutMetricData`. Los valores de métrica deben ser numéricos y estar en la raíz, y cada combinación de dimensiones crea una métrica; evita dimensiones como `requestId`.
- `CloudWatch` ofrece paneles automáticos y personalizados que pueden reunir recursos de varias regiones y, con la observabilidad entre cuentas, de varias cuentas desde una cuenta de monitorización. `Lambda Insights` recoge métricas de sistema y arranques en frío mediante una extensión, `Container Insights` hace lo mismo para ECS y EKS, y `Contributor Insights` muestra los principales contribuidores en logs.

Instrument code for observability

- El registro (logging) guarda los eventos que genera la aplicación; la monitorización evalúa de forma continua su salud y rendimiento, a menudo con métricas y alarmas obtenidas de esos registros. La observabilidad va más allá: combina métricas, registros y trazas para correlacionar señales y responder a preguntas nuevas, como el éxito de las peticiones de un grupo concreto de usuarios.
- En Lambda, con el formato JSON puedes filtrar por nivel los registros de aplicación y de sistema sin cambiar el código, aunque un nivel fijado en el propio código tiene prioridad. Varias funciones pueden escribir en un mismo grupo de registros personalizado, cuya retención define cuánto se conservan los eventos. Las políticas de protección de datos enmascaran la información sensible al inspector y solo quien tiene `logs:Unmask` la ve completa.
- Las métricas personalizadas se publican con la `API` `PutMetricData` (espacio de nombres, nombre y dimensiones) o escribiendo registros en formato de métricas incrustadas (EMF), del que `CloudWatch` extrae las métricas sin llamadas adicionales. `StorageResolution=1` crea métricas de alta resolución, y los campos `Values` y `Counts` agrupan muchos valores en una llamada conservando los percentiles que se pierden con los conjuntos estadísticos.
- En X-Ray, las anotaciones son pares clave-valor de tipo cadena, número o booleano que se indexan y permiten buscar trazas con expresiones de filtro como `annotation[clave]`; los metadatos admiten cualquier tipo, incluidos objetos, pero no se indexan. En Lambda el segmento de la función no es accesible ni modificable, así que las anotaciones se añaden en subsegmentos y `setUser` no se puede usar.
- Los cambios de estado de `CodeDeploy` y `CodePipeline` se emiten como eventos de `EventBridge`, y una regla que filtre el tipo de detalle y el estado puede enviarlos a un tema de SNS. Para las cuotas, `CloudWatch` publica métricas de uso en `AWS/Usage` y la expresión `SERVICE_QUOTA` permite alarmar al acercarse al límite. Las alarmas compuestas avisan solo cuando se cumplen varias condiciones a la vez y reducen el ruido.

Optimize applications by using AWS services and features

- En Lambda, la concurrencia es el número de peticiones en curso a la vez y se estima como peticiones por segundo por duración media en segundos. La concurrencia reservada fija un máximo y un mínimo para una función; la aprovisionada preinicializa entornos en una versión o alias, nunca en `$LATEST`. En los orígenes de eventos de `SQS`, la concurrencia máxima limita cada cola por separado.
- `CloudWatch Application Signals` recoge automáticamente métricas de volumen, disponibilidad, latencia, fallos y errores de servicios en EKS, ECS y EC2, y permite definir SLO. `Lambda Insights`, una extensión en capa, aporta métricas de sistema por invocación como tiempo de CPU, memoria, red y arranques en frío. `Amazon CodeGuru Profiler` localiza las líneas de código que más CPU consumen, y X-Ray muestra la latencia de las llamadas a `dependencias`.
- La política de filtro, cada suscriptor de un tema de SNS recibe todos los mensajes, con ella, solo los que cumplen sus condiciones. `FilterPolicyScope` decide si se evalúan los atributos del mensaje (valor predeterminado) o el cuerpo JSON. Varias propiedades se combinan con AND, varios valores de una propiedad con OR, y operadores como `numeric`, `anything-but` o `exists` afinan la coincidencia.
- Por defecto, `CloudFront` no tiene en cuenta las cabeceras al cachear; para guardar versiones distintas según una cabecera, como `Accept-Language` o `CloudFront-Is-Mobile-Viewer`, hay que incluirla en la clave de caché con una política de caché. La política de solicitud al origen envía datos al origen sin añadirlos a la clave. En `API Gateway`, las cabeceras, rutas o cadenas de consulta pueden ser claves de caché.