PassQueen
a.k.a dumpsqueen.com

AWS Certified Developer - Associate

Amazon AWS DVA-C02

Version Demo

Total Demo Questions: 62

Total Premium Questions: 628

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Deployment	113
Topic 2, Security	175
Topic 3, Development with AWS Services	262
Topic 4, Refactoring	7
Topic 5, Monitoring and Troubleshooting	71
Total	628

QUESTION NO: 1

A company has an Amazon API Gateway REST API that integrates with an AWS Lambda function. The APIâ€™s **development stage** references a Lambda **development alias** named dev.

A developer needs to make a **production alias** of the Lambda function named prod available through the API.

Which solution meets these requirements?

A. Create a new method on the API named production. Configure the method to include a stage variable that points to the prod Lambda alias.

B. Create a new method on the API named production. Configure an integration request on the development stage that points to the prod Lambda alias.

C. Deploy the API to a new stage named production. Configure the stage to include a stage variable that points to the prod Lambda alias.

D. Deploy the API to a new stage named production. Configure an integration request on the production stage that points directly to the prod Lambda alias.

ANSWER: C

Explanation:

Deploy the API to a new stage named production and configure that stage with a variable that identifies the `prod` Lambda alias. API Gateway REST API stages represent distinct runtime environments, allowing the same API deployment and method configuration to be exposed through separate stage URLs. A stage variable lets the Lambda integration resolve the appropriate backend alias for each environment, so the existing development stage can continue to invoke `dev`, while the production stage invokes `prod`.

For this pattern, the Lambda integration URI is configured to use the stage-variable value in the function ARN or alias portion of the URI. At request time, API Gateway substitutes the stage variable associated with the invoked stage. This provides an environment-specific integration without duplicating API resources and methods, and it supports a controlled promotion process: publish a Lambda version, point the `prod` alias to that version, and deploy or update the production API stage as needed. API Gateway must also have permission to invoke the qualified Lambda function ARN that includes the alias. AWS documents using stage variables with Lambda function aliases for REST API integrations and describes stages as named references to API deployments.

See API Gateway stage variables and AWS Lambda aliases.

QUESTION NO: 2

A developer creates an AWS Lambda function that is written in Java. During testing, the Lambda function does not work how the developer expected. The developer wants to use tracing capabilities to troubleshoot the problem.

Which AWS service should the developer use to accomplish this goal?

A. AWS Trusted Advisor

B. Amazon CloudWatch

C. AWS X-Ray

D. AWS CloudTrail

ANSWER: C

Explanation:

AWS X-Ray is the appropriate service because it provides distributed tracing for applications running on AWS Lambda. When active tracing is enabled for a Lambda function, Lambda sends trace data to X-Ray. This data shows the path of a request through the function and any supported downstream AWS services or instrumented application components. A

developer can use the X-Ray trace map and trace details to identify latency, errors, faults, throttling, and the specific segments or subsegments where unexpected behavior occurs. For a Java Lambda function, the developer can also instrument application code with the AWS X-Ray SDK for Java to create custom subsegments and add annotations or metadata, providing deeper visibility into important operations. This makes X-Ray well suited to troubleshooting execution flow and dependencies during testing. Lambda's tracing configuration supports active tracing, which automatically creates trace segments for invocations and integrates the function with the X-Ray service. The function's execution role must also have permissions to send trace data, commonly through the AWSXRayDaemonWriteAccess managed policy. See Using AWS X-Ray with AWS Lambda and Tracing AWS Lambda functions with X-Ray.

QUESTION NO: 3

A developer created reusable code that several AWS Lambda functions need to use. The developer bundled the code into a .zip archive. The developer needs to deploy the code to AWS and update the Lambda functions to use the code.

Which solution will meet this requirement in the MOST operationally efficient way?

A. Upload the .zip archive to Amazon S3. Configure an import path on the Lambda functions to point to the .zip archive.

B. Create a new Lambda function that contains and runs the shared code. Update the existing Lambda functions to invoke the new Lambda function synchronously.

C. Create a Lambda layer that contains the .zip archive. Attach the Lambda layer to the Lambda functions.

D. Create a Lambda container image that includes the shared code. Use the container image as a Lambda base image for all the functions.

ANSWER: C

Explanation:

Create a Lambda layer that contains the .zip archive. Attach the Lambda layer to the Lambda functions. AWS Lambda layers are designed for dependencies that are shared by multiple functions, including libraries, custom runtimes, and other reusable code. The developer publishes the archive as a layer version, then configures each applicable function to include that layer. At invocation time, Lambda extracts the layer into the function environment's `/opt` directory, where code can reference it using the language's supported dependency-search path conventions or an explicit path as appropriate.

This approach keeps shared dependencies separate from each function's deployment package, eliminating repetitive packaging and deployment of identical code. It also provides centralized versioning: functions can be migrated to a newer layer version in a controlled manner while existing versions remain available for functions that have not yet been updated. This substantially reduces ongoing maintenance when reusable code changes and is the purpose of Lambda layers. AWS documents layers as a mechanism to package libraries and other dependencies separately from function code and make them available to one or more functions. See Managing Lambda dependencies with layers and Adding layers to functions.

QUESTION NO: 4

A developer is building an application that stores objects in an Amazon S3 bucket. The bucket does not have versioning enabled. The objects are accessed rarely after 1 week. However, the objects must be immediately available at all times.

The developer wants to optimize storage costs for the S3 bucket.

Which solution will meet this requirement?

A. Create an S3 Lifecycle rule to expire objects after 7 days.

B. Create an S3 Lifecycle rule to transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 7 days.

C. Create an S3 Lifecycle rule to transition objects to S3 Glacier Flexible Retrieval after 7 days.

D. Create an S3 Lifecycle rule to delete objects that have delete markers.

E. Create an S3 Lifecycle rule to transition objects to S3 Intelligent-Tiering after 7 days.

ANSWER: E

Explanation:

Create an S3 Lifecycle rule to transition objects to S3 Intelligent-Tiering after 7 days. This class is designed for data with changing or uncertain access patterns while preserving millisecond access to objects. Objects initially reside in the Frequent Access tier. If they are not accessed for 30 consecutive days, S3 automatically moves them to the Infrequent Access tier, which has a lower storage price and still provides immediate retrieval. S3 Intelligent-Tiering can also move eligible objects to optional archive tiers, but those tiers should not be enabled here because the requirement is for immediate availability at all times.

This approach is necessary because S3 Standard-IA has a 30-day minimum storage duration. Consequently, an object cannot be transitioned from S3 Standard to S3 Standard-IA after only 7 days through a lifecycle policy. By transitioning to S3 Intelligent-Tiering after one week instead, the application remains compliant with the requested timing, and S3 subsequently applies the lower-cost infrequent-access tier when the object's lack of access meets the 30-day threshold. See the Amazon S3 storage class documentation and S3 Intelligent-Tiering overview.

QUESTION NO: 5

A developer at a company recently created a serverless application to process and show data from business reports. The application ' s user interface (UI) allows users to select and start processing the files. The Ul displays a message when the result is available to view. The application uses AWS Step Functions with AWS Lambda functions to process the files. The developer used Amazon API Gateway and Lambda functions to create an API to support the UI.

The company ' s Ul team reports that the request to process a file is often returning timeout errors because of the see or complexity of the files. The Ul team wants the API to provide an immediate response so that the Ul can deploy a message while the files are being processed. The backend process that is invoked by the API needs to send an email message when the report processing is complete.

What should the developer do to configure the API to meet these requirements?

A. Change the API Gateway route to add an X-Amz-Invocation-Type header win a sialic value of ' Event ' in the integration request Deploy the API Gateway stage to apply the changes.

B. Change the configuration of the Lambda function that implements the request to process a file. Configure the maximum age of the event so that the Lambda function will ion asynchronously.

C. Change the API Gateway timeout value to match the Lambda function ominous value. Deploy the API Gateway stage to apply the changes.

D. Change the API Gateway route to add an X-Amz-Target header with a static value of ' A sync ' in the integration request Deploy me API Gateway stage to apply the changes.

ANSWER: A

Explanation:

Change the API Gateway route to add an X-Amz-Invocation-Type header win a sialic value of ' Event ' in the integration request Deploy the API Gateway stage to apply the changes. This configures API Gateway to invoke the integrated Lambda function asynchronously. The `X-Amz-Invocation-Type: Event` header tells the Lambda Invoke API to accept the event for asynchronous processing rather than hold the HTTP request open until the function finishes. API Gateway can therefore return an immediate successful response to the UI, allowing the UI to show that report processing is underway instead of encountering a timeout for lengthy or complex files.

The asynchronously invoked Lambda function can start the Step Functions workflow, or the processing workflow can otherwise continue independently after the API response has been sent. When the report-processing workflow reaches completion, its completion logic can send the required email notification, such as by invoking a Lambda function that uses Amazon SES. This cleanly separates the short-lived API request from the potentially long-running backend workflow. AWS documents configuring asynchronous Lambda invocation through API Gateway by adding the static invocation-type header in the integration request. See Invoking a Lambda function asynchronously with API Gateway and AWS Lambda asynchronous invocation.

QUESTION NO: 6

A company uses more than 100 AWS Lambda functions to handle application services. One Lambda function is critical and must always run successfully. The company notices that occasionally, the critical Lambda function does not initiate. The company investigates the issue and discovers instances of the Lambda TooManyRequestsException: Rate Exceeded error in Amazon CloudWatch logs. Upon further review of the logs, the company notices that some of the non-critical functions run properly while the critical function fails. A developer must resolve the errors and ensure that the critical Lambda function runs successfully. Which solution will meet these requirements with the LEAST operational overhead?

A. Configure reserved concurrency for the critical Lambda function. Set reserved concurrent executions to the appropriate level.

B. Configure provisioned concurrency for the critical Lambda function. Set provisioned concurrent executions to the appropriate level.

C. Configure CloudWatch alarms for TooManyRequestsException errors. Add the critical Lambda function as an alarm state change action to invoke the critical function again after a failure.

D. Configure CloudWatch alarms for TooManyRequestsException errors. Add Amazon EventBridge as an action for the alarm state change. Use EventBridge to invoke the critical function again after a failure.

ANSWER: A

Explanation:

Configure reserved concurrency for the critical Lambda function. Set reserved concurrent executions to the appropriate level. Reserved concurrency assigns a specified portion of the AWS account's available Lambda concurrency exclusively to that function. This prevents the function from being starved of execution capacity when numerous non-critical functions consume the shared, unreserved concurrency pool. The observed `TooManyRequestsException: Rate Exceeded` condition is consistent with Lambda throttling caused by insufficient available account concurrency. By choosing a reserved concurrency value that covers the critical function's expected peak simultaneous invocations, the company ensures that this capacity remains available for the critical workload regardless of demand from other functions. Reserved concurrency also imposes a maximum concurrency for the function, so the selected value should be sized using expected traffic and execution duration. This is a direct Lambda configuration change, requires no retry workflow, monitoring automation, or additional event-processing resources, and therefore has the least operational overhead while addressing the underlying concurrency contention. AWS documents that reserved concurrency both reserves concurrency for a function and removes that amount from the unreserved pool used by other functions. See AWS Lambda concurrency configuration and AWS Lambda scaling and concurrency.

QUESTION NO: 7

A developer creates an AWS Lambda function to publish a message to an Amazon SNS topic. All message content must be encrypted in transit and at rest between Lambda and Amazon SNS. A part of the Lambda execution role is as follows:

" Effect " : " Allow " , " Action " : " SNS:Publish " , " Resource " : " arn:aws:sns:us-east-1:1234567890:secure-topic "

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

A. Enable server-side encryption on the SNS topic.

B. Add a Deny statement to the Lambda execution role. Specify the SNS topic ARN as the resource. Specify " aws:SecureTransport " : " true " as the condition.

C. Create a VPC endpoint for Amazon SNS.

D. Add a StringEquals condition of " sns:Protocol " : " https " to the Lambda execution role.

E. Add a Deny statement to the Lambda execution role. Specify the SNS topic ARN as the resource. Specify " aws:SecureTransport " : " false " as the condition.

ANSWER: A E

Explanation:

Enabling server-side encryption on the SNS topic protects message content at rest. Amazon SNS uses AWS Key Management Service keys for this capability, encrypting message data while it is stored by the service. The topic can use an AWS managed key or a customer managed KMS key, according to the organization's key-management requirements. This setting directly addresses the at-rest portion of the requirement for the secure topic.

Adding a Deny statement to the Lambda execution role with `aws:SecureTransport` set to `false` enforces encryption in transit. `aws:SecureTransport` is an AWS global condition key that indicates whether the request used SSL/TLS. An explicit deny for requests where this value is false prevents the role from publishing through an unencrypted connection, regardless of the existing allow for `SNS:Publish`. In an IAM policy, this condition is typically expressed using the `Bool` condition operator. Together, SNS server-side encryption and a transport-security deny ensure that the Lambda role can publish only over TLS while SNS stores the message encrypted at rest. See the Amazon SNS server-side encryption documentation and the aws:SecureTransport IAM condition-key documentation.

QUESTION NO: 8

A company stores audit records in an Amazon S3 bucket. The records must be retained for 7 years and must not be overwritten or deleted during the retention period, including by users with administrative permissions. The company needs to use a managed S3 feature.

Which actions should the developer take? (Choose two.)

A. Create the S3 bucket with S3 Object Lock enabled.

B. Set a 7-year retention period in compliance mode for the audit object versions.

C. Enable an S3 Lifecycle rule that transitions the objects to the S3 Glacier Flexible Retrieval storage class.

D. Configure a bucket policy that grants the audit administrators full access to the objects.

E. Enable S3 Cross-Region Replication to a second bucket.

ANSWER: A B

Explanation:

Create the S3 bucket with S3 Object Lock enabled. Object Lock can be enabled only when a bucket is created, and it requires S3 Versioning. Object Lock protects individual object versions from being deleted or overwritten for a defined retention period.

Set a retention period in compliance mode for the audit object versions. In compliance mode, protected object versions cannot be overwritten or deleted by any user, including the AWS account root user, until the retention period expires. This meets the requirement to protect records from administrative deletion during the full retention period. See the Amazon S3 Object Lock documentation and the Amazon S3 Object Lock retention modes documentation.

QUESTION NO: 9

A company has an application that uses an Amazon S3 bucket for object storage. A developer needs to configure in-transit encryption for the S3 bucket. All the S3 objects containing personal data needs to be encrypted at rest with AWS KMS keys, which can be rotated on demand.

Which combination of steps will meet these requirements? (Select TWO.)

A. Write an S3 bucket policy to allow only encrypted connections over HTTPS by using permissions boundary.

B. Configure an S3 bucket policy to enable client-side encryption for the objects containing personal data by using an AWS KMS customer managed key

C. Configure the application to encrypt the objects by using an AWS KMS customer managed key before uploading the objects containing personal data to Amazon S3.

D. Write an S3 bucket policy to allow only encrypted connections over HTTPS by using the aws:SecureTransport condition.

E. Configure S3 Block Public Access settings for the S3 bucket to allow only encrypted connections over HTTPS.

ANSWER: C D

Explanation:

"Write an S3 bucket policy to allow only encrypted connections over HTTPS by using the aws:SecureTransport condition" enforces encryption in transit. The `aws:SecureTransport` global condition key identifies whether a request uses SSL/TLS. A bucket policy can explicitly deny S3 actions when this key is `false`, ensuring that clients must use HTTPS when accessing objects or performing other bucket operations. This policy-based control applies regardless of the caller, helping protect personal data while it travels between the application and Amazon S3.

"Configure the application to encrypt the objects by using an AWS KMS customer managed key before uploading the objects containing personal data to Amazon S3" provides encryption at rest before the data reaches the bucket. With client-side encryption integrated with AWS KMS, the application can use a customer managed KMS key to protect data-encryption keys and upload ciphertext to S3. Customer managed keys provide key-management control, including the ability to initiate on-demand rotation for supported symmetric KMS keys. Thus, the data remains encrypted in S3 and the company retains control over the KMS key lifecycle. See the Amazon S3 policy condition keys documentation and the AWS KMS key rotation documentation.

QUESTION NO: 10

A company has an application that is deployed on AWS Elastic Beanstalk. The application generates user-specific PDFs and stores the PDFs in an Amazon S3 bucket. The application then uses Amazon Simple Email Service (Amazon SES) to send the PDFs by email to subscribers.

Users no longer access the PDFs 90 days after the PDFs are generated. The S3 bucket is not versioned and contains many obsolete PDFs.

A developer must reduce the number of files in the S3 bucket by removing PDFs that are older than 90 days.

Which solution will meet this requirement with the LEAST development effort?

A. Update the application code. In the code, add a rule to scan all the objects in the S3 bucket every day and to delete objects after 90 days.

B. Create an AWS Lambda function. Program the Lambda function to scan all the objects in the S3 bucket every day and to delete objects after 90 days.

C. Create an S3 Lifecycle rule for the S3 bucket to expire objects after 90 days.

D. Partition the S3 objects with a < year > / < month > / < day > key prefix. Create an AWS Lambda function to remove objects that have prefixes that have reached the expiration date.

ANSWER: C

Explanation:

Create an S3 Lifecycle rule for the S3 bucket to expire objects after 90 days. An Amazon S3 Lifecycle expiration rule is a managed, declarative way to automatically delete current object versions after a specified age. Because the bucket is not versioned, each PDF is a single current object, and S3 can permanently remove it when it reaches 90 days after creation. The rule can apply to the entire bucket or be limited with a prefix or object tags if the bucket contains files that must be retained. This approach requires only lifecycle configuration; it does not require application changes, scheduled compute, custom scanning logic, pagination over bucket contents, or operational maintenance for deletion jobs. S3 evaluates lifecycle rules asynchronously, so expiration indicates that objects become eligible for removal at the specified age rather than guaranteeing deletion at an exact time of day. This behavior still meets the requirement to remove obsolete PDFs automatically with the least development effort. For configuration details, see the Amazon S3 Lifecycle management documentation and the S3 lifecycle expiration considerations.

QUESTION NO: 11

A financial services company builds a credit card transaction processing application that uses an Amazon API Gateway **HTTP API** and AWS Lambda functions. The application logs all requests and request parameters to Amazon CloudWatch. The application makes the logs accessible to developer AWS accounts and a separate fraud detection AWS account by using a cross-account **IAM role** .

The company requires that only the fraud detection account be able to view customer credit card numbers that are associated with the transactions. Developers at the company must not be able to use the credit card numbers for testing or debugging.

The developers create the following data protection policy document snippet:

{

" Name " : " data-protection-policy " ,

" Description " : " Credit card redaction " ,

" Version " : " 2021-06-01 " ,

" Statement " : [{

" Sid " : " redact-policy " ,

" DataIdentifier " : [

" arn:aws:dataprotection::aws:data-identifier/CreditCardNumber "

],

" Operation " : {

" Deidentify " : {

" MaskConfig " : {}

}

}

}]

}

Which combination of actions must the developers take to comply with the new policy? (Select TWO.)

A. Add an **UnmaskConfig** property to the **Operation** property of the data protection policy. Specify the role that the fraud detection account must assume.

B. Add the **logs:Unmask** permission to the IAM role that the fraud detection account must assume.

C. Add the data protection policy to the CloudWatch log group that captures logs for the HTTP API.

D. Add the data protection policy to the CloudWatch log group in the account that hosts the application.

E. Add the data protection policy to the IAM role that the fraud detection account must assume.

ANSWER: B C

Explanation:

Adding the **logs:Unmask** permission to the IAM role that the fraud detection account must assume gives that specifically authorized cross-account role the ability to retrieve protected log data in unmasked form. CloudWatch Logs data protection masks matching sensitive data by default, including values detected by the managed `CreditCardNumber` data identifier. The `logs:Unmask` permission is the additional IAM authorization required when an identity needs to reveal those protected

values. Access can therefore remain restricted to the fraud detection role, while developer roles do not receive that permission.

Adding the data protection policy to the CloudWatch log group that captures logs for the HTTP API enables the policy's de-identification operation for the actual incoming log events containing request parameters. CloudWatch Logs data protection policies are configured on log groups and evaluate log events that are ingested after the policy is applied. The supplied `MaskConfig` directs CloudWatch Logs to mask detected card numbers, preventing ordinary log viewers from seeing the underlying values. Together, log-group masking and narrowly granted unmask authorization satisfy the requirement for protected developer access and privileged fraud-investigation access. See the CloudWatch Logs data protection documentation and the CloudWatch Logs permissions reference.

QUESTION NO: 12

A company is running a custom web application on Amazon EC2 instances behind an Application Load Balancer. The instances run in an Auto Scaling group. The company ' s development team is using AWS CloudFormation to deploy all the services. The application is time-consuming to install and configure when the development team launches a new instance.

Which combination of steps should a developer take to optimize the performance when a new instance is launched? (Select TWO.)

A. Use an AWS Marketplace Amazon Machine Image (AMI) with a prebuilt application.

B. Create a prebuilt Amazon Machine Image (AMI) with the application installed and configured.

C. Update the launch template resource in the CloudFormation template.

D. Use AWS Systems Manager Run Command to install and configure the application.

E. Use CloudFormation helper scripts to install and configure the application.

ANSWER: B C

Explanation:

Create a prebuilt Amazon Machine Image (AMI) with the application installed and configured. is correct because an AMI can contain the application, its dependencies, and its required baseline configuration before an Auto Scaling event occurs. Instances launched from that image avoid performing lengthy installation and configuration work during boot. This reduces the time before an instance is ready to respond to Application Load Balancer health checks and receive traffic, while also making launch behavior more consistent. AWS describes an AMI as a template that contains the software configuration required to launch an instance; teams commonly use custom AMIs as immutable, preconfigured deployment artifacts.

Update the launch template resource in the CloudFormation template. is also correct because the Auto Scaling group must be configured to launch instances from the newly created custom AMI. Updating the CloudFormation-managed launch template to specify the custom AMI ID makes the optimized instance configuration repeatable and version controlled. New instances launched by the Auto Scaling group will then inherit the baked application image automatically, rather than requiring manual setup or runtime installation steps. For details, see the AWS documentation on Amazon Machine Images and AWS::EC2::LaunchTemplate.

QUESTION NO: 13

A developer is deploying a new version of an AWS Lambda function. The function is invoked through a Lambda alias that is used by a production Amazon API Gateway API. The developer needs to shift a small percentage of traffic to the new version before shifting all traffic. The deployment must automatically roll back if Amazon CloudWatch detects an elevated error rate.

Which actions should the developer take? (Choose two.)

A. Create a CodeDeploy deployment group that uses the Lambda alias and a canary or linear deployment configuration.

B. Configure a CloudWatch alarm in the CodeDeploy deployment group and enable automatic rollback.

C. Update the API Gateway integration to invoke the $LATEST version of the Lambda function.

D. Create a separate Lambda function for each percentage of production traffic.

E. Increase the reserved concurrency of the new Lambda function version before deployment.

ANSWER: A B

Explanation:

Create an AWS CodeDeploy deployment group that uses the Lambda alias and a canary or linear deployment configuration. CodeDeploy can update a Lambda alias to gradually shift invocation traffic between the currently deployed function version and a new published version. The API Gateway integration can continue invoking the stable alias while CodeDeploy manages weighted routing behind that alias.

The developer should also configure a CloudWatch alarm in the CodeDeploy deployment group and enable automatic rollback. If the error-rate metric breaches the alarm threshold during the traffic shift, CodeDeploy stops the deployment and restores the alias to the previously deployed version. This provides controlled rollout and automated recovery without custom traffic-routing code. See the AWS CodeDeploy deployment configurations and AWS CodeDeploy monitoring with CloudWatch alarms.

QUESTION NO: 14

An application is processing clickstream data using Amazon Kinesis. The clickstream data feed into Kinesis experiences periodic spikes. The PutRecords API call occasionally fails and the logs show that the failed call returns the response shown below:



Which techniques will help mitigate this exception? (Choose two.)

A. Increase the number of shards in the Kinesis data stream.

B. Use a PutRecord API instead of PutRecords.

C. Reduce the frequency and/or size of the requests.

D. Use Amazon SNS instead of Kinesis.

E. Reduce the number of KCL consumers.

ANSWER: A C

Explanation:

Increasing the number of shards in the Kinesis data stream increases the stream's aggregate write capacity. Each shard supports up to 1,000 records per second and up to 1 MiB per second of write throughput. Adding shards therefore provides additional capacity to absorb periodic clickstream spikes and reduces the likelihood that a `PutRecords` request will encounter a provisioned-throughput exception.

Reducing the frequency and/or size of the requests also helps keep writes within the per-shard throughput limits. Producers should control batching and request rates so that records routed to an individual shard do not exceed its record-rate or byte-rate capacity. In production, the producer should also retry records that receive a throughput-related error using exponential backoff; because `PutRecords` can partially succeed, retries should target only the records identified as failed in the response. These capacity and producer-rate controls directly mitigate transient spikes while preserving the use of Kinesis for ordered, partition-key-based stream ingestion. See the Amazon Kinesis Data Streams quotas and limits and the PutRecords API reference.

QUESTION NO: 15

A company is using Amazon RDS as the Backend database for its application. After a recent marketing campaign, a surge of read requests to the database increased the latency of data retrieval from the database.

The company has decided to implement a caching layer in front of the database. The cached content must be encrypted and must be highly available.

Which solution will meet these requirements?

A. Amazon Cloudfront

B. Amazon ElastiCache to Memcached

C. Amazon ElastiCache for Redis in cluster mode

D. Amazon DynamoDB Accelerate (DAX)

ANSWER: C

Explanation:

Amazon ElastiCache for Redis in cluster mode meets the requirements because it provides an in-memory cache that an application can use to serve frequently requested database results, reducing read pressure and retrieval latency for Amazon RDS. ElastiCache for Redis supports encryption in transit using TLS and encryption at rest for supported replication groups, protecting cached data while it is transmitted and stored. Encryption settings are configured when the replication group is created and integrate with AWS Key Management Service for encryption at rest.

Cluster mode distributes data across multiple shards, allowing the cache tier to scale horizontally as read demand grows. For high availability, shards can have replica nodes deployed across Availability Zones. ElastiCache monitors the primary node and can automatically promote a replica if a primary node fails, helping the application continue serving cached reads with minimal disruption. This combination of encrypted storage and transport, sharding, replicas, and automatic failover makes it appropriate as a resilient caching layer in front of the RDS backend. See the ElastiCache encryption documentation and the ElastiCache replication and failover documentation.

QUESTION NO: 16

An organization is using Amazon CloudFront to ensure that its users experience low-latency access to its web application. The organization has identified a need to encrypt all traffic between users and CloudFront, and all traffic between CloudFront and the web application.

How can these requirements be met? (Select TWO)

A. Use AWS KMS to encrypt traffic between CloudFront and the web application.

B. Set the Origin Protocol Policy to "HTTPS Only".

C. Set the origin's HTTP port to 443.

D. Set the Viewer Protocol Policy to "HTTPS Only" or "Redirect HTTP to HTTPS".

E. Enable the CloudFront option Restrict Viewer Access.

ANSWER: B D

Explanation:

Setting the Origin Protocol Policy to “HTTPS Only” ensures that CloudFront uses HTTPS for every request it sends from an edge location to the origin web application. This protects the origin-facing connection with TLS encryption. The origin must therefore support HTTPS and present a certificate that CloudFront can validate. AWS documents the HTTPS Only origin protocol policy as the setting that causes CloudFront to communicate with the origin only over HTTPS.

Setting the Viewer Protocol Policy to “HTTPS Only” prevents CloudFront from serving content over an unencrypted viewer connection. Alternatively, “Redirect HTTP to HTTPS” accepts an initial HTTP request only to return a redirect, after which the viewer accesses the content through HTTPS. Both settings ensure that actual content delivery between the user and CloudFront occurs over HTTPS. A CloudFront distribution also needs an appropriate viewer certificate configured for its alternate domain name when a custom domain is used. Together, these two protocol-policy settings encrypt both required network segments: viewer-to-CloudFront and CloudFront-to-origin. See the CloudFront documentation for HTTPS communication with origins and the documentation for requiring HTTPS between viewers and CloudFront.

QUESTION NO: 17

A company has on-premises data centers that run an image processing service. The service consists of containerized applications that run on Kubernetes clusters. All the applications have access to the same NFS share for files and data storage. The company is running out of NFS capacity in the data centers and needs to migrate to AWS as soon as possible. The Kubernetes clusters must be highly available on AWS. Which combination of actions will meet these requirements? (Select TWO.)

A. Transfer the information that is in the NFS share to an Amazon EBS volume. Upload the container images to Amazon ECR.

B. Transfer the information that is in the NFS share to an Amazon EFS volume. Upload the container images to Amazon ECR.

C. Create an Amazon ECS cluster to run the applications. Configure each node of the cluster to mount the Amazon EBS volume at the required path for the container images.

D. Create an Amazon EKS cluster to run the applications. Configure each node of the cluster to mount the Amazon EBS volume at the required path for the container images.

E. Create an Amazon EKS cluster to run the applications. Configure each node of the cluster to mount the Amazon EFS volume at the required path for the container images.

ANSWER: B E

Explanation:

“Transfer the information that is in the NFS share to an Amazon EFS volume. Upload the container images to Amazon ECR.” is correct because Amazon EFS provides managed, elastic shared file storage that is accessed by using the NFS protocol. It can be mounted concurrently by multiple Kubernetes worker nodes and is therefore an appropriate AWS replacement for the common on-premises NFS share. Amazon ECR is AWS’s managed container image registry, providing a compatible destination for the application images that the migrated workloads will pull.

“Create an Amazon EKS cluster to run the applications. Configure each node of the cluster to mount the Amazon EFS volume at the required path for the container images.” is correct because Amazon EKS preserves the existing Kubernetes deployment model, minimizing application and operational changes during an urgent migration. EKS provides a managed Kubernetes control plane that operates across multiple Availability Zones for high availability. EFS is a Regional service and can provide shared access through mount targets in the Availability Zones used by the worker nodes. In practice, the EFS mount would be presented to containers at the required application data path, while the container images are retrieved from ECR. Together, ECR, EFS, and a multi-AZ EKS deployment provide a managed, highly available migration architecture. See Amazon EFS User Guide and Amazon EKS User Guide.

QUESTION NO: 18

A company is creating an application that processes csv files from Amazon S3 A developer has created an S3 bucket The developer has also created an AWS Lambda function to process the csv files from the S3 bucket

Which combination of steps will invoke the Lambda function when a csv file is uploaded to Amazon S3? (Select TWO.)

A. Create an Amazon EventBridge rule Configure the rule with a pattern to match the S3 object created event

B. Schedule an Amazon EventBridge rule to run a new Lambda function to scan the S3 bucket.

C. Add a trigger to the existing Lambda function. Set the trigger type to EventBridge Select the Amazon EventBridge rule.

D. Create a new Lambda function to scan the S3 bucket for recently added S3 objects

E. Add S3 Lifecycle rules to invoke the existing Lambda function

ANSWER: A C

Explanation:

"Create an Amazon EventBridge rule Configure the rule with a pattern to match the S3 object created event" is required because Amazon S3 can send object events to Amazon EventBridge when EventBridge integration is enabled for the bucket. An EventBridge event pattern can match S3 Object Created events and can be made more specific, such as matching the target bucket and object keys that end in `.csv`. This ensures that an upload produces an event eligible to invoke the processor.

"Add a trigger to the existing Lambda function. Set the trigger type to EventBridge Select the Amazon EventBridge rule." completes the integration by associating the Lambda function as the target of that EventBridge rule. When an incoming S3 event matches the rule, EventBridge asynchronously invokes the selected Lambda function and supplies the event details, including bucket and object-key information, so the function can locate and process the uploaded CSV file. Lambda resource-based permissions are configured to allow EventBridge to invoke the function when the target is added through the AWS console or corresponding AWS APIs. See the Amazon S3 EventBridge integration documentation and the EventBridge documentation for Lambda targets.

QUESTION NO: 19

A company has a serverless application that uses an Amazon API Gateway API to invoke an AWS Lambda function. A developer creates a fix for a defect in the Lambda function code. The developer wants to deploy this fix to the production environment. To test the changes, the developer needs to send 10% of the live production traffic to the updated Lambda function version.

Options:

A. A. Publish a new version of the Lambda function that contains the updated code.

B. B. Set up a new stage in API Gateway with a new Lambda function version. Enable weighted routing in API Gateway stages.

C. C. Create an alias for the Lambda function. Configure weighted routing on the alias. Specify a 10% weight for the new Lambda function version.

D. D. Set up a routing policy on a Network Load Balancer. Configure 10% of the traffic to go to the new Lambda function version.

E. E. Set up a weighted routing policy by using Amazon Route 53. Configure 10% of the traffic to go to the new Lambda function version.

ANSWER: A C

Explanation:

Publishing a new version of the Lambda function that contains the updated code is required because Lambda versions are immutable snapshots of a function's code and configuration. Weighted traffic shifting is supported only between published function versions; the mutable `$LATEST` version cannot be used as either target of alias routing. The existing production version remains the primary version, and the newly published version supplies the tested code snapshot.

Create an alias for the Lambda function. Configure weighted routing on the alias. Specify a 10% weight for the new Lambda function version. A Lambda alias normally points to one function version, but its routing configuration can assign an additional version a decimal weight from 0 through 1. Setting the new version's additional weight to `0.1` directs approximately 10% of alias-invoked requests to that version while approximately 90% continue to use the primary production version. API Gateway should invoke the production Lambda alias so the distribution applies to live API requests. This enables a controlled canary deployment while retaining a stable version for most traffic.

For implementation details, see the AWS Lambda documentation on function aliases and weighted alias routing.

QUESTION NO: 20

A developer needs to modify an application architecture to meet new functional requirements. Application data is stored in Amazon DynamoDB and processed tor analysis in a nightly batch. The system analysts do not want to wait until the next day to view the processed data and have asked to have it available in near-real time.

Which application architecture pattern would enable the data to be processed as it is received?

A. Event driven

B. Client-server d riven

C. Fan-out driven

D. Schedule driven

ANSWER: A

Explanation:

Event driven is correct because it enables processing to begin in response to each data change, rather than waiting for a scheduled nightly batch. Amazon DynamoDB can publish item-level changes through DynamoDB Streams. A consumer, commonly an AWS Lambda function, can be invoked when stream records are available and can transform, aggregate, analyze, or persist the updated data for analysts with near-real-time availability. DynamoDB Streams records modifications in the order they occur for each item, providing an effective source of events for downstream processing. Lambda integrates directly with DynamoDB Streams through an event source mapping that polls stream shards and invokes the function with batches of change records. This decouples writes to the operational DynamoDB table from analytical processing, while allowing the processing workflow to react automatically as new data is received. The resulting latency is typically seconds rather than the delay imposed by a once-per-day batch job. For implementation details, see the Amazon DynamoDB Streams Developer Guide and Using AWS Lambda with Amazon DynamoDB.

QUESTION NO: 21

A company uses an AWS Lambda function to transfer files from an Amazon S3 bucket to the company ' s SFTP server. The Lambda function connects to the SFTP server by using credentials such as username and password. The company uses Lambda environment variables to store these credentials.

A developer needs to implement encrypted username and password credentials.

Which solution will meet these requirements?

A. Remove the user credentials from the Lambda environment. Implement 1AM database authentication.

B. Move the user credentials from Lambda environment variables to AWS Systems Manager Parameter Store.

C. Move the user credentials from Lambda environment variables to AWS Key Management Service (AWS KMS).

D. Move the user credentials from the Lambda environment to an encrypted .txt file. Store the file in an S3 bucket.

ANSWER: B

Explanation:

Move the user credentials from Lambda environment variables to AWS Systems Manager Parameter Store. Parameter Store is designed to store configuration data and secrets centrally, and the developer can create the username and password as `SecureString` parameters. SecureString values are encrypted with AWS Key Management Service (AWS KMS), using either the AWS managed key for Systems Manager Parameter Store or a customer managed KMS key. This prevents the plaintext credentials from being stored directly in application configuration.

The Lambda function should retrieve the parameters at runtime by using the Systems Manager Parameter Store API or AWS SDK with decryption enabled, such as `GetParameter` with `WithDecryption=true`. Its execution role must be granted narrowly scoped permissions to read the specific parameter ARNs and, when a customer managed key is used, permission to decrypt with that KMS key. This approach separates secrets from function deployment configuration, encrypts the stored values, and enables centralized access control and auditing through IAM, KMS, and CloudTrail. AWS documents SecureString as the Parameter Store type for sensitive data that must be encrypted at rest. See SecureString parameters in Parameter Store and the GetParameter API reference.

QUESTION NO: 22

A mobile app stores blog posts in an Amazon DynacnoDB table Millions of posts are added every day and each post represents a single item in the table. The mobile app requires only recent posts. Any post that is older than 48 hours can be removed.

What is the MOST cost-effective way to delete posts that are older man 48 hours?

A. For each item add a new attribute of type String that has a timestamp that is set to the blog post creation time. Create a script to find old posts with a table scan and remove posts that are order than 48 hours by using the Balch Write Item API operation. Schedule a cron job on an Amazon EC2 instance once an hour to start the script.

B. For each item add a new attribute of type. String that has a timestamp that its set to the blog post creation time. Create a script to find old posts with a table scan and remove posts that are Oder than 48 hours by using the Batch Write item API operating. Place the script in a container image. Schedule an Amazon Elastic Container Service (Amazon ECS) task on AWS Far gate that invokes the container every 5 minutes.

C. For each item, add a new attribute of type Date that has a timestamp that is set to 48 hours after the blog post creation time. Create a global secondary index (GSI) that uses the new attribute as a sort key. Create an AWS Lambda function that references the GSI and removes expired items by using the Batch Write item API operation Schedule me function with an Amazon CloudWatch event every minute.

D. For each item add a new attribute of type. Number that has timestamp that is set to 48 hours after the blog post. creation time Configure the DynamoDB table with a TTL that references the new attribute.

ANSWER: D

Explanation:

For each item add a new attribute of type. Number that has timestamp that is set to 48 hours after the blog post. creation time Configure the DynamoDB table with a TTL that references the new attribute. This uses Amazon DynamoDB Time to Live (TTL), the managed expiration mechanism designed for automatically removing items that are no longer needed. The TTL attribute must be stored as a DynamoDB Number containing the expiration time in Unix epoch time, expressed in seconds. For each blog post, the application calculates an expiry timestamp equal to its creation time plus 48 hours and writes that value with the item. After TTL is enabled for that attribute, DynamoDB identifies items whose expiration time has passed and deletes them automatically, without application-side scans, scheduled compute, or explicit delete requests. This is the most cost-effective approach at a scale of millions of new items per day because DynamoDB does not charge for TTL deletions. Expired items are typically removed within a few days of their expiration timestamp, so TTL is appropriate where the requirement permits asynchronous physical removal rather than deletion at an exact minute. The application should still use its normal read patterns to retrieve recent posts and, if necessary, can filter out items based on the expiration attribute during the TTL deletion window. See the DynamoDB TTL developer guide and TTL requirements and best practices.

QUESTION NO: 23

A company's website runs on an Amazon EC2 instance and uses Auto Scaling to scale the environment during peak times. Website users across the world ate experiencing high latency flue lo sialic content on theEC2 instance. even during non-peak hours.

When companion of steps mill resolves the latency issue? (Select TWO)

A. Double the Auto Scaling group's maximum number of servers

B. Host the application code on AWS Lambda

C. Scale vertically by resizing the EC2 instances

D. Create an Amazon CloudFront distribution to cache the static content

E. Store the application's static content in Amazon S3

ANSWER: D E

Explanation:

Creating an Amazon CloudFront distribution to cache the static content and storing the application's static content in Amazon S3 addresses the actual source of the latency: globally distributed users are retrieving static assets from a single EC2-based origin. CloudFront is a content delivery network that serves cached objects from edge locations close to viewers. Requests for cacheable assets such as images, stylesheets, JavaScript files, and downloads can therefore avoid repeated round trips to the EC2 instance's AWS Region. This reduces network latency for users worldwide and also decreases load on the application servers.

Amazon S3 is an appropriate durable, highly available origin for these static assets. CloudFront can use an S3 bucket as its origin, retrieve an object when it is not already cached, and then cache it at edge locations according to the configured cache behavior and object headers. The application can continue to run dynamic functionality on EC2 while static files are delivered efficiently through CloudFront. For secure production access, the bucket can remain private and CloudFront can be granted access through Origin Access Control. See the Amazon CloudFront Developer Guide and Using Amazon S3 with CloudFront.

QUESTION NO: 24

A company uses AWS X-Ray to monitor a serverless application. The components of the application have different request rates. The user interactions and transactions are important to trace, but they are low in volume. The background processes such as application health checks, polling, and connection maintenance generate high volumes of read-only requests.

Currently, the default X-Ray sampling rules are universal for all requests. Only the first request per second and some additional requests are recorded. This setup is not helping the company review the requests based on service or request type.

A developer must configure rules to trace requests based on service or request properties. The developer must trace the user interactions and transactions without wasting effort recording minor background tasks.

Which solution will meet these requirements?

A. Disable sampling for high-volume read-only requests. Sample at a lower rate for all requests that handle user interactions or transactions.

B. Disable sampling and trace all requests for requests that handle user interactions or transactions. Sample high-volume read-only requests at a higher rate.

C. Disable sampling and trace all requests for requests that handle user interactions or transactions. Sample high-volume read-only requests at a lower rate.

D. Disable sampling for high-volume read-only requests. Sample at a higher rate for all requests that handle user interactions or transactions.

ANSWER: C

Explanation:

Disable sampling and trace all requests for requests that handle user interactions or transactions. Sample high-volume read-only requests at a lower rate. meets the requirement because AWS X-Ray supports custom sampling rules that match requests using properties such as the service name, host, HTTP method, URL path, and resource ARN. The developer can create a high-priority rule for the low-volume, business-critical interaction and transaction endpoints with a sampling rate of 100%, ensuring that every relevant request is traced. A separate rule can match health checks, polling, and connection-maintenance traffic and apply a low sampling rate, substantially reducing trace volume for routine read-only operations. X-Ray evaluates sampling rules by priority, so more specific rules for critical request types should be prioritized ahead of broader rules. This approach provides complete observability for important user activity while controlling telemetry volume and associated processing effort for frequent background traffic. AWS X-Ray sampling rules use a reservoir and fixed rate to determine which requests are recorded; configuring rules by request characteristics is specifically intended for workloads with different tracing needs. See the AWS X-Ray sampling rules documentation and the AWS X-Ray sampling behavior guide.

QUESTION NO: 25

A developer is building an application that needs to store an API key. An AWS Lambda function needs to use the API key. The developer ' s company requires secrets to be encrypted at rest by an AWS KMS key. The company must control key rotation.

Which solutions will meet these requirements? (Select TWO.)

A. Store the API key as an AWS Secrets Manager secret. Encrypt the secret with an AWS managed KMS key.

B. Store the API key as an AWS Systems Manager Parameter Store String parameter.

C. Store the API key as an AWS Systems Manager Parameter Store SecureString parameter. Encrypt the parameter with a customer managed KMS key.

D. Store the API key in a Lambda environment variable. Encrypt the environment variable with an AWS managed KMS key.

E. Store the API key in a Lambda environment variable. Encrypt the environment variable with a customer managed KMS key.

ANSWER: C E

Explanation:

Storing the API key as an AWS Systems Manager Parameter Store SecureString parameter and encrypting it with a customer managed KMS key meets the requirements because SecureString parameters are intended for sensitive values and are encrypted with KMS. The company can administer the customer managed key, including its key policy, permissions, enabled state, and rotation configuration. The Lambda execution role can be granted permission to retrieve the parameter and to decrypt it with that specific key. This supports centralized secret storage while applying least-privilege access controls.

Storing the API key in a Lambda environment variable encrypted with a customer managed KMS key also meets the requirements. Lambda encrypts environment variables at rest, and a function configuration can specify a customer managed KMS key rather than relying on the default AWS managed key. The organization therefore retains administrative control of the encryption key and can configure automatic rotation or perform on-demand rotation as required. The function can access the configured environment-variable value during execution, while KMS permissions and the key policy protect its decryption. For both approaches, a customer managed KMS key is the essential feature that gives the company control over its rotation and governance lifecycle. See the AWS Systems Manager SecureString documentation and AWS Lambda environment-variable encryption documentation.

QUESTION NO: 26

A developer is using an AWS account to build an application that stores files in an Amazon S3 bucket. Files must be encrypted at rest by AWS KMS keys. A second AWS account must have access to read files from the bucket.

The developer wants to minimize operational overhead for the application.

Which combination of solutions will meet these requirements? (Select TWO.)

A. Use a customer managed key to encrypt the files. Create a key policy that grants kms: Decrypt permissions to the second AWS account.

B. Use an AWS managed key to encrypt the files. Create a key policy that grants kms:Decrypt permissions to the second AWS account.

C. Create a service control policy (SCP) that grants s3:GetObject permissions to the second AWS account.

D. Create a bucket policy for the S3 bucket that grants s3:GetObject permissions to the second AWS account.

E. Create a gateway endpoint for the S3 bucket. Modify the endpoint policy to grant s3:GetObject permissions to the second AWS account.

ANSWER: A D

Explanation:

Use a customer managed key to encrypt the files. Create a key policy that grants kms:Decrypt permissions to the second AWS account. A customer managed KMS key supports a key policy that permits cross-account use. The key policy can authorize the second account, and IAM permissions in that account can then be assigned to the specific role or user that reads the objects. For Amazon S3 objects encrypted with SSE-KMS, the reader needs KMS decrypt permission in addition to S3 object-read permission. Customer managed keys provide this needed cross-account control while AWS operates the underlying key infrastructure.

Create a bucket policy for the S3 bucket that grants s3:GetObject permissions to the second AWS account. An S3 bucket policy is a resource-based policy and is the standard low-overhead mechanism for authorizing a principal in another AWS account to retrieve objects. Together, the bucket policy authorizes access to the encrypted S3 objects and the customer managed key policy authorizes the KMS cryptographic operation required to transparently decrypt them during retrieval. This avoids application-managed encryption keys, key distribution, or custom credential workflows. For implementation details, see the Amazon S3 cross-account access walkthrough and the AWS KMS key policy documentation.

QUESTION NO: 27

When a developer tries to run an AWS Code Build project, it raises an error because the length of all environment variables exceeds the limit for the combined maximum of characters.

What is the recommended solution?

A. Add the export LC-_ALL " on _ US, tuft " command to the pre _ build section to ensure POSIX Localization.

B. Use Amazon Cognate to store key-value pairs for large numbers of environment variables

C. Update the settings for the build project to use an Amazon S3 bucket for large numbers of environment variables

D. Use AWS Systems Manager Parameter Store to store large numbers ot environment variables

ANSWER: D

Explanation:

Use AWS Systems Manager Parameter Store to store large numbers ot environment variables. CodeBuild imposes a combined character limit on environment variables defined directly in a build project, so moving configuration values out of the project environment definition avoids concentrating all values in that constrained field. Parameter Store is designed for centralized storage of configuration data and secrets, with support for hierarchical parameter names, encryption through SecureString parameters, IAM-based access control, and auditability. A build can retrieve the required values at runtime by declaring Parameter Store names in the buildspec file's `env.parameter-store` section. CodeBuild then resolves those parameter names and makes the retrieved values available to build commands as environment variables. This lets the project retain only the parameter references in source-controlled build configuration while the actual configuration is managed separately and can be updated without embedding values in the CodeBuild project. The CodeBuild service role

must be granted permission to retrieve the relevant parameters, such as `ssm:GetParameters`, and, for encrypted SecureString values using a customer managed KMS key, permission to decrypt with that key. See the CodeBuild buildspec Parameter Store environment-variable reference and the AWS Systems Manager Parameter Store documentation.

QUESTION NO: 28

A developer compiles an AWS Lambda function and packages the result as a .zip file. The developer uses the Functions page on the Lambda console to attempt to upload the local packaged .zip file. When pushing the package to Lambda, the console returns the following error:

Which solutions can the developer use to publish the code? (Select TWO.)

A. Upload the package to Amazon S3. Use the Functions page on the Lambda console to upload the package from the S3 location.

B. Create an AWS Support ticket to increase the maximum package size.

C. Use the update-function-code AWS CLI command. Pass the -publish parameter.

D. Repackage the Lambda function as a Docker container image. Upload the image to Amazon Elastic Container Registry {Amazon ECR). Create a new Lambda function by using the Lambda console. Reference the image that is deployed to Amazon ECR.

E. Sign the .zip file digitally. Create a new Lambda function by using the Lambda console. Update the configuration of the new Lambda function to include the Amazon Resource Name (ARN) of the code signing configuration.

ANSWER: A D

Explanation:

Uploading the package to Amazon S3 and then specifying that S3 object as the Lambda function's code source is an appropriate solution when a deployment package cannot be uploaded directly through the console. Lambda limits direct ZIP uploads to 50 MB, whereas a ZIP package supplied from Amazon S3 can be up to 250 MB after decompression, including layers. The S3 bucket must be in the same AWS Region as the Lambda function, and Lambda retrieves the referenced deployment artifact during the code update or function creation process.

Repackaging the function as a Docker container image and publishing it to Amazon Elastic Container Registry (Amazon ECR) is also a supported Lambda deployment model. Lambda can create a function from an ECR image URI, enabling a substantially larger package size: a container image can be up to 10 GB. The image must be stored in an Amazon ECR repository in the same Region as the Lambda function. This approach is particularly useful when application dependencies make the ZIP deployment artifact too large. See the AWS Lambda quotas documentation and Deploy Lambda functions with container images.

QUESTION NO: 29

A company is creating an AWS Step Functions state machine to run a set of tests for an application. The tests need to run when a specific AWS Cloud Formation stack is deployed.

Which combination of steps will meet these requirements? (Select TWO.)

A. Create an AWS Lambda function to invoke the state machine.

B. Create an Amazon EventBridge rule on the default bus that matches on a detail type of CloudFormation stack status change, a status of CREATE_COMPLETE or UPDATE_COMPLETE, and the stack ID of the CloudFormation stack.

C. Create a pipe in Amazon EventBridge Pipes that has a source of the default event bus. Set the Lambda function as a target. Filter on a detail type of CloudFormation stack status change, a status of UPDATE_IN_PROGRESS, and the stack ID of the CloudFormation stack.

D. Create a pipe in Amazon EventBridge Pipes that has a source of the EventBridge rule. Set the state machine as a target.

E. Add the state machine as a target of the EventBridge rule.

ANSWER: B E

Explanation:

Create an Amazon EventBridge rule on the default event bus that matches the CloudFormation stack-status-change event for the required stack and a completed deployment status. CloudFormation sends stack status change events to EventBridge, and the event pattern can filter on the stack ID in the event detail. To cover both initial deployments and subsequent stack deployments, the rule can match `CREATE_COMPLETE` and `UPDATE_COMPLETE`. Matching a completion status ensures that the test workflow begins only after CloudFormation has successfully finished the applicable stack operation.

Add the state machine as a target of the EventBridge rule. EventBridge supports AWS Step Functions state machines as rule targets and starts a state machine execution when an event matches. This provides a direct event-driven integration, without requiring application code to poll CloudFormation or an intermediary compute service. The EventBridge target must have an IAM role that permits `states:StartExecution` for the selected state machine. The event can also be supplied as the execution input, allowing the workflow to identify the stack and deployment event that initiated the tests. See the CloudFormation events in Amazon EventBridge documentation and Amazon EventBridge targets documentation.

QUESTION NO: 30

A developer at a company has created a containerized application to serve public reporting APIs. The developer needs to migrate the application to run by using an Amazon ECS service. However, the company does not want to manage the underlying hosts that run the application. The developer must be able to monitor CPU and memory utilization of the application by using the Amazon CloudWatch console. Which combination of actions will meet these requirements? (Select TWO.)

A. Configure the ECS cluster instances to use an EC2 user data script. Set the Amazon ECS agent ECS_DISABLE_METRICS environment variable to false.

B. Write a script to publish a custom CloudWatch alarm for CPU utilization and memory utilization.

C. Configure the task definition to use an IAM role that has allow permissions for the cloudwatch:GetMetricData action.

D. Specify FARGATE in the requiresCompatibilities field of the task definition.

E. Specify limits for the CPU field and the memory field of the task definition.

ANSWER: D E

Explanation:

Specify FARGATE in the requiresCompatibilities field of the task definition. meets the requirement to run the Amazon ECS service without the company managing underlying hosts. AWS Fargate is the serverless compute engine for Amazon ECS: AWS provisions and operates the infrastructure that runs the tasks, while the developer defines the task and service configuration. A Fargate-compatible task definition must declare `FARGATE` in `requiresCompatibilities`.

Specify limits for the CPU field and the memory field of the task definition. is also required for Fargate tasks. At the task level, Fargate requires valid CPU and memory values that define the compute and memory resources allocated to each running task. Amazon ECS publishes service-level CPU and memory utilization metrics to the `AWS/ECS` CloudWatch namespace. These metrics express utilization as a percentage of the CPU and memory resources reserved by the tasks in the service, making the configured task resources essential to meaningful utilization monitoring in the CloudWatch console.

This combination provides a serverless ECS deployment model and the resource definitions needed for ECS service utilization metrics. See the AWS documentation for AWS Fargate and Amazon ECS CloudWatch metrics.

QUESTION NO: 31

A company is using AWS CloudFormation to deploy a two-tier application. The application will use Amazon RDS as its backend database. The company wants a solution that will randomly generate the database password during deployment. The solution also must automatically rotate the database password without requiring changes to the application.

What is the MOST operationally efficient solution that meets these requirements ' ?

A. Use an AWS Lambda function as a CloudFormation custom resource to generate and rotate the password.

B. Use an AWS Systems Manager Parameter Store resource with the SecureString data type to generate and rotate the password.

C. Use a cron daemon on the application s host to generate and rotate the password.

D. Use an AWS Secrets Manager resource to generate and rotate the password.

ANSWER: D

Explanation:

Use an AWS Secrets Manager resource to generate and rotate the password. AWS Secrets Manager integrates directly with AWS CloudFormation through the `AWS::SecretsManager::Secret` resource. Its `GenerateSecretString` property can generate a random database password during stack deployment, avoiding hardcoded credentials or custom password-generation code. The secret can be associated with the Amazon RDS database and used through a dynamic reference or retrieved by the application at runtime.

Secrets Manager also provides managed, scheduled credential rotation. A CloudFormation `AWS::SecretsManager::RotationSchedule` resource configures how frequently rotation occurs. For supported Amazon RDS engines, Secrets Manager provides managed rotation workflows that update both the database credential and the stored secret. Because the application retrieves the current secret value rather than embedding a fixed password in its configuration, it can continue operating after rotations without application code changes. This is the most operationally efficient design because CloudFormation provisions the secret and rotation configuration declaratively, while Secrets Manager handles secure storage, random generation, and ongoing rotation. See the AWS::SecretsManager::Secret CloudFormation reference and the AWS Secrets Manager rotation documentation.

QUESTION NO: 32

A company is using the AWS Serverless Application Model (AWS SAM) to develop a social media application. A developer needs a quick way to test AWS Lambda functions locally by using test event payloads. The developer needs the structure of these test event payloads to match the actual events that AWS services create.

A. Create shareable test Lambda events. Use these test Lambda events for local testing.

B. Store manually created test event payloads locally. Use the sam local invoke command with the file path to the payloads.

C. Store manually created test event payloads in an Amazon S3 bucket. Use the sam local invoke command with the S3 path to the payloads.

D. Use the sam local generate-event command to create test payloads for local testing.

ANSWER: D

Explanation:

Use the `sam local generate-event` command to create test payloads for local testing. AWS SAM CLI provides this command specifically to generate sample event JSON for event sources that invoke Lambda functions, including services such as Amazon S3, Amazon API Gateway, Amazon SNS, Amazon SQS, EventBridge, and DynamoDB. The generated payload follows the event schema that the selected AWS service uses when invoking a Lambda function, so the developer does not need to handcraft the nested JSON structure or guess required fields.

The developer can generate an event for a particular service and event type, save or pipe the JSON output to a file if desired, and then supply it to `sam local invoke`. For example, a generated Amazon S3 notification event can be passed directly into a local Lambda invocation, enabling rapid validation of parsing, routing, and business logic before deployment. This is especially useful when a function supports several AWS event sources because each source has a distinct payload format. AWS documents the available event-source templates and usage in the AWS SAM CLI generate-event documentation. Local invocation behavior and event input are further described in the AWS SAM local invoke documentation.

QUESTION NO: 33

A company invokes an AWS Lambda function asynchronously to process orders. The function calls a third-party service that occasionally experiences extended outages. The company needs to preserve events that the function cannot process after all retry attempts are exhausted. The company also needs to limit the amount of time that Lambda continues to retry failed events.

Which actions should the developer take? (Choose two.)

A. Configure a maximum event age for the Lambda function.

B. Configure an on-failure destination for the Lambda function.

C. Configure provisioned concurrency for the Lambda function.

D. Increase the Lambda function memory allocation.

E. Configure an Amazon SQS queue as an event source for the Lambda function.

ANSWER: A B

Explanation:

Configure a maximum event age for the Lambda function. For asynchronous invocations, Lambda can continue retrying an event until the event expires. Configuring the maximum event age limits how long Lambda retains and retries an event after the original invocation.

Configure an on-failure destination for the Lambda function. When Lambda cannot successfully process an asynchronous event after the configured retry and event-age settings are exhausted, Lambda sends an invocation record to the configured destination. An Amazon SQS queue is an appropriate destination for retaining failed invocation records for later investigation or reprocessing. See Handling errors for asynchronous Lambda invocations.

QUESTION NO: 34

A developer is debugging an issue with an application that is based on an AWS Lambda function. The Lambda function intermittently fails during a 1-hour window. The developer needs to investigate the root cause of the intermittent failures. The application sends logs to an Amazon CloudWatch Logs log group. The developer must be able to collect logs that relate to Lambda function failures. The developer must capture the dates and times at which the failures occurred. Which solution will meet these requirements in the MOST operationally efficient way?

A. Use the AWS CLI in AWS CloudShell to browse the CloudWatch Logs log group to search for the error messages.

B. Use CloudWatch Logs Insights to run a query on the log group that searches for specific patterns that match the error messages.

C. Download log files from the CloudWatch Logs log group to the developer ' s local machine. Use a local text search tool to search for the error messages.

D. Export the CloudWatch Logs log group to an Amazon S3 bucket. Use Amazon Athena to run a SQL query against the bucket to search for the error messages.

ANSWER: B

Explanation:

Use CloudWatch Logs Insights to run a query on the log group that searches for specific patterns that match the error messages. CloudWatch Logs Insights is designed for interactive analysis of data already stored in CloudWatch Logs, so the developer can select the relevant Lambda log group, restrict the query to the known 1-hour failure window, and filter messages for exceptions, errors, timeouts, or other failure indicators. Each matching log event includes the `@timestamp` field, allowing the developer to directly identify and record the date and time of each failure. For example, a query can filter on error patterns and display `@timestamp`, `@message`, and `@logStream`, then sort chronologically to correlate failures with Lambda invocation details. This approach requires no log export, additional storage configuration, data pipeline, or local

tooling. It provides targeted troubleshooting against the existing operational logs and is therefore the most operationally efficient solution for investigating an intermittent issue in a bounded time period. AWS documents that Logs Insights supports querying log groups, choosing a time range, filtering fields, and displaying timestamped results. See the Amazon CloudWatch Logs Insights documentation and the AWS Lambda logging documentation.

QUESTION NO: 35

A developer is working on a web application that runs on Amazon ECS and uses an Amazon DynamoDB table to store data. The application performs a large number of read requests against a small set of the table data.

How can the developer improve the performance of these requests? (Select TWO.)

A. Create an Amazon ElastiCache cluster. Configure the application to cache data in the cluster.

B. Create a DynamoDB Accelerator (DAX) cluster. Configure the application to use the DAX cluster for DynamoDB requests.

C. Configure the application to make strongly consistent read requests against the DynamoDB table.

D. Increase the read capacity of the DynamoDB table.

E. Enable DynamoDB adaptive capacity.

ANSWER: A B

Explanation:

Creating an Amazon ElastiCache cluster and configuring the application to cache data in the cluster is an effective solution because the workload repeatedly reads a small, frequently accessed data set. The ECS application can retrieve commonly requested values from an in-memory Redis or Memcached cache rather than querying DynamoDB for every request. This reduces application read latency and decreases the volume of requests that must reach the DynamoDB table. The application can implement an appropriate cache-aside strategy and expiration policy to ensure cached values remain suitable for its data-freshness requirements.

Creating a DynamoDB Accelerator (DAX) cluster and configuring the application to use the DAX cluster for DynamoDB requests is also designed specifically for this access pattern. DAX is a fully managed, in-memory caching service for DynamoDB that is compatible with DynamoDB API calls. It caches frequently requested items and query results, allowing repeated reads to be served with microsecond latency in many cases while offloading read activity from DynamoDB. DAX is particularly convenient when the application primarily needs DynamoDB-oriented caching without building its own cache management logic. AWS documents DAX as a read-through cache that can improve read performance for eventually consistent workloads. See Amazon DynamoDB Accelerator (DAX) and Amazon ElastiCache documentation.

QUESTION NO: 36

A developer configures an AWS Lambda function to run in private subnets in a VPC. The function must call a public third-party HTTPS API. The VPC does not have an existing internet egress solution.

Which actions should the developer take to allow the function to call the third-party API? (Choose two.)

A. Create a NAT gateway in a public subnet and associate an Elastic IP address with the NAT gateway.

B. Update the route table for the Lambda function private subnets to route internet-bound traffic to the NAT gateway.

C. Assign a public IPv4 address directly to the Lambda function.

D. Create an internet gateway and attach the internet gateway directly to each private subnet.

E. Create a gateway VPC endpoint for Amazon S3.

ANSWER: A B

Explanation:

Create a NAT gateway in a public subnet and assign an Elastic IP address to the NAT gateway. A NAT gateway provides outbound IPv4 connectivity for resources in private subnets while preventing unsolicited inbound connections from the internet. The public subnet route table must also have a route to an internet gateway so that the NAT gateway can reach public internet endpoints.

Update the route table associated with the Lambda function private subnets to route internet-bound traffic, such as `0.0.0.0/0`, to the NAT gateway. Lambda functions configured for VPC access use the route tables of their selected subnets. Merely assigning a function to a private subnet does not provide internet access. A public IP address cannot be assigned directly to Lambda network interfaces to solve this requirement. See the AWS Lambda documentation for internet access from a VPC and the Amazon VPC NAT gateway documentation.

QUESTION NO: 37

A company maintains a REST service using Amazon API Gateway and the API Gateway native API key validation. The company recently launched a new registration page, which allows users to sign up for the service. The registration page creates a new API key using CreateApiKey and sends the new key to the user. When the user attempts to call the API using this key. the user receives a 403 Forbidden error. Existing users are unaffected and can still call the API.

What code updates will grant these new users access to the API?

A. The createDeploymer.t method must be called so the API can be redeployed to include the newly created API key.

B. The updateAuthorizer method must be called to update the API ' s authorizer to include the newly created API key

C. The importApiKeys method must be called to import all newly created API keys into the current stage of the API.

D. The createUsagePlanKey method must be called to associate the newly created API key with the correct usage plan.

ANSWER: D

Explanation:

The createUsagePlanKey method must be called to associate the newly created API key with the correct usage plan. Creating an API key creates a credential, but does not by itself authorize that credential to invoke methods in an API stage that requires an API key. For REST APIs in API Gateway, API keys are made usable by associating them with a usage plan. The usage plan defines the API stages that the key can access and can also apply throttling and quota limits. Existing users continue to work because their API keys are already associated with the applicable usage plan. The registration workflow should therefore create the key and then call `CreateUsagePlanKey`, supplying the target usage plan ID and the new API key ID. Once this association is created, the new user can send the key in the `x-api-key` request header to access methods configured to require API keys. No API redeployment is needed merely to add a key to an existing usage plan, because the stage-to-usage-plan association is already configured. See the API Gateway CreateUsagePlanKey API reference and the API Gateway usage plans documentation.

QUESTION NO: 38

A developer is migrating a containerized application from an on-premises environment to the AWS Cloud. The developer is using the AWS CDK to provision a container in Amazon ECS on AWS Fargate. The container is behind an Application Load Balancer (ALB).

When the developer deploys the stack, the deployment fails because the ALB fails health checks. The developer needs to resolve the failed health checks.

Which solutions will meet this requirement? (Select TWO.)

A. Confirm that the capacity providers for the container have been provisioned and are properly sized.

B. Confirm that the target group port matches the port mappings in the ECS task definition.

C. Confirm that a hosted zone associated with the ALB matches a hosted zone that is referenced in the ECS task definition.

D. Confirm that the ALB listener on the mapped port has a default action that redirects to the application ' s health check path endpoint.

E. Confirm that the ALB listener on the mapped port has a default action that forwards to the correct target group.

ANSWER: B E

Explanation:

Confirming that the target group port matches the port mappings in the ECS task definition ensures that the Application Load Balancer can reach the application process that is listening inside each Fargate task. For an Amazon ECS service integrated with an Application Load Balancer, the service load-balancer configuration identifies the container name and container port to register as targets. The target group must therefore be configured consistently with the application's network and port configuration so that health-check requests reach a listening endpoint.

Confirming that the ALB listener on the mapped port has a default action that forwards to the correct target group ensures that requests arriving at the listener are routed to the target group containing the ECS service tasks. The listener, target group, and ECS service must be associated correctly as part of the load-balanced service architecture provisioned by CDK. Once the correct target group is used and its registered task targets are reachable on the application port, the target group can evaluate the configured health-check path and mark responsive tasks healthy.

AWS documents the ECS service load-balancer configuration, including the required container name and container port association, at Amazon ECS service load balancing. Listener forwarding behavior is described in the Application Load Balancer listener documentation.

QUESTION NO: 39

A developer owns and supports an application that has database credentials that are stored in environment variables for AWS Lambda functions. The developer needs an alternate storage method for the credentials instead of environment variables in plaintext. Which solution will handle the credentials MOST securely?

A. Store the database credentials as a secret in AWS Secrets Manager. Set the secret ' s Amazon Resource Name (ARN) as the value of an environment variable. Use the AWS Parameters and Secrets Lambda Extension to retrieve the credentials in the Lambda function.

B. Use base64 encoding for the database credentials. Include the database credentials into the Lambda function ' s source code as part of a build process. Update the Lambda function ' s code to decode the credentials before the credentials are used.

C. Store the database credentials as a string type parameter in the AWS Systems Manager Parameter Store. Set the parameter ' s Amazon Resource Name (ARN) as the value of an environment variable. Use the AWS Parameters and Secrets Lambda Extension to retrieve the credentials from the Lambda function.

D. Use AWS CloudFormation to deploy the application. Ensure that the NoEcho property is set to true for the parameters that contain the database credentials.

ANSWER: A

Explanation:

Store the database credentials as a secret in AWS Secrets Manager. Set the secret's Amazon Resource Name (ARN) as the value of an environment variable. Use the AWS Parameters and Secrets Lambda Extension to retrieve the credentials in the Lambda function. This approach places the sensitive credential value in AWS Secrets Manager, a purpose-built managed service for storing, controlling access to, and rotating secrets such as database usernames and passwords. The Lambda environment variable contains only the secret identifier (the ARN), not the plaintext secret itself. The function's execution role can be granted least-privilege permission to retrieve only that specific secret, using permissions such as `secretsmanager:GetSecretValue`. The AWS Parameters and Secrets Lambda Extension retrieves the secret locally through its endpoint and caches it within the Lambda execution environment. This reduces repeated calls to Secrets Manager, helping lower latency and API-call costs while retaining centralized secret management. Secrets Manager also supports automatic rotation for supported database engines and rotation workflows, allowing applications to use refreshed credentials without embedding values in deployment artifacts or function code. See the AWS Secrets Manager User Guide and the AWS Lambda documentation for retrieving secrets with the extension.

QUESTION NO: 40

A company recently deployed an AWS Lambda function. A developer notices an increase in the function throttle metrics in Amazon CloudWatch.

What are the MOST operationally efficient solutions to reduce the function throttling? (Select TWO.)

A. Migrate the function to Amazon EKS.

B. Increase the maximum age of events in Lambda.

C. Increase the function's reserved concurrency.

D. Add the lambda:GetFunctionConcurrency action to the execution role.

E. Request a service quota change for increased concurrency.

ANSWER: C E

Explanation:

Increasing the function's reserved concurrency is an efficient remedy when the function has a reserved concurrency setting that is limiting the number of simultaneous executions. Reserved concurrency both reserves capacity for a function and establishes its maximum concurrent executions. Raising that value, while ensuring sufficient unreserved account concurrency remains available, lets Lambda run more invocations in parallel and can directly reduce the `Throttles` metric.

Requesting a service quota change for increased concurrency addresses throttling caused by the AWS account's regional Lambda concurrency quota. Lambda applies concurrency controls at both the function and account level. If aggregate concurrent demand reaches the account limit, Lambda throttles additional invocations even where a particular function does not have an overly restrictive reserved concurrency setting. A quota increase supplies additional account-level capacity, allowing the workload to scale to a higher level of concurrent execution.

These actions target the concurrency capacity that governs Lambda throttling and can be implemented without redesigning or migrating the application. AWS recommends monitoring concurrency and throttle metrics to determine whether the constraint is a function reservation or the regional account quota. See AWS Lambda concurrency configuration and Lambda quotas.

QUESTION NO: 41

A developer must securely access a secret during a build process in an AWS CodeBuild project that has an IAM role. The secret must remain encrypted at rest and must be passed to the buildspec.yml file without appearing in the build logs. Which solution will meet these requirements with the LEAST operational overhead?

A. Configure AWS Secrets Manager to store the secret. Configure the env section of the buildspec.yml file to have a secrets-manager key-value entry that references the secret. Grant the CodeBuild IAM role the minimum necessary permissions to access the secret.

B. Store the secret in an encrypted Amazon S3 bucket. Download the secret during the build process. Use NoEcho to mask the value of the secret. Grant the CodeBuild IAM role the minimum necessary permissions to access the S3 bucket.

C. Configure AWS Systems Manager Parameter Store to store the secret. Configure the env section of the buildspec.yml file to have a parameter-store key-value entry that references the parameter. Grant the CodeBuild IAM role the minimum necessary permissions to access the parameter.

D. Store the secret in AWS Systems Manager Parameter Store. Use a pre-build step to retrieve the parameter by using AWS CLI commands. Mask the parameter in the buildspec.yml file. Grant the CodeBuild IAM role the minimum necessary permissions to access the parameter.

ANSWER: A

Explanation:

Configure AWS Secrets Manager to store the secret. Configure the env section of the buildspec.yml file to have a secrets-manager key-value entry that references the secret. Grant the CodeBuild IAM role the minimum necessary permissions to access the secret. This is the lowest-overhead solution because CodeBuild directly supports Secrets Manager references in the `env/secrets-manager` section of a buildspec. At build startup, CodeBuild retrieves the specified secret value and makes it available to the build environment without requiring custom retrieval scripts, temporary files, or separate secret-distribution logic. Secrets Manager encrypts secrets at rest, using an AWS managed key by default or a customer managed AWS KMS key when configured. Access remains controlled through the CodeBuild service role, which should receive only the required `secretsmanager:GetSecretValue` permission and, when applicable, permission to use the KMS key. CodeBuild documentation specifically supports mapping an environment-variable name to a Secrets Manager secret and states that secrets should not be written to build output; the build commands must also avoid explicitly printing the variable. This native integration provides managed encryption, IAM-based authorization, and buildspec-level injection with minimal implementation and maintenance work. See the CodeBuild buildspec secrets-manager reference and the AWS Secrets Manager documentation.

QUESTION NO: 42

A company has an analytics application that uses an AWS Lambda function to process transaction data asynchronously A developer notices that asynchronous invocations of the Lambda function sometimes fail When failed Lambda function invocations occur, the developer wants to invoke a second Lambda function to handle errors and log details.

Which solution will meet these requirements?

A. Configure a Lambda function destination with a failure condition Specify Lambda function as the destination type Specify the error-handling Lambda function ' s Amazon Resource Name (ARN) as the resource

B. Enable AWS X-Ray active tracing on the initial Lambda function. Configure X-Ray to capture stack traces of the failed invocations. Invoke the error-handling Lambda function by including the stack traces in the event object.

C. Configure a Lambda function trigger with a failure condition Specify Lambda function as the destination type Specify the error-handling Lambda function ' s Amazon Resource Name (ARN) as the resource

D. Create a status check alarm on the initial Lambda function. Configure the alarm to invoke the error-handling Lambda function when the alarm is initiated. Ensure that the alarm passes the stack trace in the event object.

ANSWER: A

Explanation:

Configure a Lambda function destination with a failure condition. Specify Lambda function as the destination type. Specify the error-handling Lambda function's Amazon Resource Name (ARN) as the resource. Lambda destinations are designed for asynchronous invocation outcomes and can route an invocation record when processing fails after Lambda has completed its configured retry behavior. An `on-failure` destination can target another Lambda function, allowing a dedicated error-handling function to receive the failed event context and execution-result metadata. The destination payload includes useful details such as the original request payload, response context, function ARN, request ID, timestamp, retry attempts, error code, and error message. The error-handling function can then log these details or take custom remediation actions without adding error-routing logic to the primary transaction-processing function. The source function’s execution role must have permission to invoke the destination function, and the destination function must allow invocation as required by its resource-based policy. This is a native Lambda asynchronous invocation feature and directly fulfills the need to invoke another Lambda function specifically when asynchronous processing fails. See the AWS Lambda documentation for invocation destinations and asynchronous invocation error handling.

QUESTION NO: 43

In a move toward using microservices, a company’s management team has asked all development teams to build their services so that API requests depend only on that service’s data store. One team is building a Payments service which has its own database; the service needs data that originates in the Accounts database. Both are using Amazon DynamoDB.

What approach will result in the simplest, decoupled, and reliable method to get near-real time updates from the Accounts database?

A. Use AWS Glue to perform frequent ETL updates from the Accounts database to the Payments database.

B. Use Amazon ElastiCache in Payments, with the cache updated by triggers in the Accounts database.

C. Use Amazon Data Firehose to deliver all changes from the Accounts database to the Payments database.

D. Use Amazon DynamoDB Streams to deliver all changes from the Accounts database to the Payments database.

ANSWER: D

Explanation:

Use Amazon DynamoDB Streams to deliver all changes from the Accounts database to the Payments database. DynamoDB Streams provides a managed, time-ordered change data capture feed for item-level inserts, updates, and deletes in the Accounts table. The Payments service can consume those records, typically through an AWS Lambda event source mapping, and update its own DynamoDB table with only the account information it needs. This preserves the microservices boundary: Payments reads from its own datastore during API requests rather than querying Accounts synchronously.

The approach is near real time because stream records are made available promptly after writes occur. It is also reliable because DynamoDB Streams integrates directly with DynamoDB and retains stream records for up to 24 hours, while Lambda can process batches and retry failed processing. The consumer should be designed to be idempotent, since a stream-processing workflow can deliver a record more than once. This event-driven replication pattern keeps the Accounts service independent of Payments while allowing Payments to maintain a local, service-owned projection of Account data.

See the Amazon DynamoDB Streams Developer Guide and Using AWS Lambda with DynamoDB Streams.

QUESTION NO: 44

An Amazon Kinesis Data Firehose delivery stream is receiving customer data that contains personally identifiable information. A developer needs to remove pattern-based customer identifiers from the data and store the modified data in an Amazon S3 bucket.

What should the developer do to meet these requirements?

A. Implement Kinesis Data Firehose data transformation as an AWS Lambda function. Configure the function to remove the customer identifiers. Set an Amazon S3 bucket as the destination of the delivery stream.

B. Launch an Amazon EC2 instance. Set the EC2 instance as the destination of the delivery stream. Run an application on the EC2 instance to remove the customer identifiers. Store the transformed data in an Amazon S3 bucket.

C. Create an Amazon OpenSearch Service instance. Set the OpenSearch Service instance as the destination of the delivery stream. Use search and replace to remove the customer identifiers. Export the data to an Amazon S3 bucket.

D. Create an AWS Step Functions workflow to remove the customer identifiers. As the last step in the workflow, store the transformed data in an Amazon S3 bucket. Set the workflow as the destination of the delivery stream.

ANSWER: A

Explanation:

Implement Kinesis Data Firehose data transformation as an AWS Lambda function. Configure the function to remove the customer identifiers. Set an Amazon S3 bucket as the destination of the delivery stream. Kinesis Data Firehose supports invoking an AWS Lambda function to transform records while they are being delivered. The function receives records from Firehose, decodes and processes each record, and returns the transformed payload for Firehose to buffer and deliver to its configured destination. This provides a managed, event-driven way to apply custom pattern matching and redaction logic, such as regular expressions that identify customer identifiers, before data is written to Amazon S3. The delivery stream remains responsible for batching, buffering, retries, scaling, and S3 delivery, while Lambda contains only the application-specific sanitization code. The Lambda execution role must have the appropriate permissions, and the Firehose delivery stream must be permitted to invoke the function. The transformation function should return valid records in the Firehose response format so the redacted content can proceed through the delivery pipeline. This architecture directly satisfies both requirements: identifiers are removed in transit and only the modified data is delivered to the S3 bucket. See the Amazon Data Firehose data transformation documentation and the AWS Lambda integration guide for Amazon Data Firehose.

QUESTION NO: 45

A company runs an application as an Amazon ECS service on an AWS Fargate cluster. The company has configured the service to use the rolling update deployment type. The company set the minimum healthy percentage to 100% and set the maximum healthy percentage to 200%.

A developer observes that a new deployment of the ECS service is failing continuously. The developer must ensure that Amazon ECS restores the most recently completed service deployment when a new service deployment fails.

Which solution will meet this requirement?

A. Use Amazon ECS service auto scaling. Set the scaling policy type as step scaling.

B. Use Amazon ECS service auto scaling. Set the scaling policy type as target tracking.

C. Use the blue/green deployment type. Enable the deployment circuit breaker and rollback on failures options.

D. Use the rolling deployment type. Enable deployment circuit breaker and rollback on failures options.

ANSWER: D

Explanation:

Use the rolling deployment type. Enable deployment circuit breaker and rollback on failures options. The Amazon ECS deployment circuit breaker is designed for services that use the rolling update deployment type (`ECS`). It monitors whether the new deployment can reach a completed, steady state. For example, it detects when replacement tasks repeatedly cannot start or fail load balancer, container, or task health checks before the deployment becomes healthy. Enabling the rollback setting instructs Amazon ECS to automatically roll the service back when the circuit breaker marks the new deployment as failed. ECS selects the most recent deployment that reached the `COMPLETED` state and uses it as the rollback deployment, restoring the previously stable service revision without requiring a manual update. The configured minimum and maximum healthy percentages can remain in place; they control how ECS replaces tasks during the rolling deployment, while the circuit breaker and rollback settings supply the required failed-deployment recovery behavior. This feature is supported for ECS rolling deployments on AWS Fargate. For configuration details and rollback behavior, see the Amazon ECS deployment circuit breaker documentation and the Amazon ECS service deployment parameters documentation.

QUESTION NO: 46

A developer is creating an application that uses an AWS Lambda function to transform and load data from an Amazon S3 bucket. When the developer tests the application, the developer finds that some invocations of the Lambda function are slower than others.

The developer needs to update the Lambda function to have predictable invocation durations that run with low latency. Any initialization activities, such as loading libraries and instantiating clients, must run during allocation time rather than during actual function invocations.

Which combination of steps will meet these requirements? (Select TWO.)

A. Create a schedule group in Amazon EventBridge Scheduler to invoke the Lambda function.

B. Configure provisioned concurrency for the Lambda function to have the necessary number of execution environments.

C. Use the $LATEST version of the Lambda function.

D. Configure reserved concurrency for the Lambda function to have the necessary number of execution environments.

E. Deploy changes, and publish a new version of the Lambda function.

ANSWER: B E

Explanation:

Configure provisioned concurrency for the Lambda function to have the necessary number of execution environments and deploy changes, then publish a new version of the Lambda function. Provisioned concurrency pre-initializes a specified number of Lambda execution environments before requests arrive. During this allocation and initialization process, Lambda runs initialization code outside the request path, including code that loads libraries, establishes SDK clients, and initializes other resources defined outside the handler. As a result, requests that are served by those provisioned environments avoid cold-start initialization latency and have more predictable invocation durations.

Provisioned concurrency can be configured only for a published function version or an alias that points to a published version. Therefore, after making the required code changes, the developer must publish a new version and configure provisioned concurrency on that immutable version or an alias targeting it. This ensures Lambda allocates and keeps the requested number of initialized environments ready for traffic. The provisioned concurrency value should be sized for the expected concurrent request volume so that requests remain within the pre-initialized capacity. For details, see the AWS Lambda provisioned concurrency documentation and the AWS Lambda versions documentation.

QUESTION NO: 47

A company has an Amazon S3 bucket that contains sensitive data. The data must be encrypted in transit and at rest. The company encrypts the data in the S3 bucket by using an AWS Key Management Service (AWS KMS) key. A developer needs to grant several other AWS accounts the permission to use the S3 GetObject operation to retrieve the data from the S3 bucket.

How can the developer enforce that all requests to retrieve the data provide encryption in transit?

A. Define a resource-based policy on the S3 bucket to deny access when a request meets the condition “aws:SecureTransport”: “false”.

B. Define a resource-based policy on the S3 bucket to allow access when a request meets the condition “aws:SecureTransport”: “false”.

C. Define a role-based policy on the other accounts' roles to deny access when a request meets the condition of “aws:SecureTransport”: “false”.

D. Define a resource-based policy on the KMS key to deny access when a request meets the condition of “aws:SecureTransport”: “false”.

ANSWER: A

Explanation:

Define a resource-based policy on the S3 bucket to deny access when a request meets the condition “aws:SecureTransport”: “false”. This is the appropriate enforcement point because Amazon S3 evaluates the bucket policy for every request to the bucket and its objects, including cross-account `s3:GetObject` requests. The global condition key `aws:SecureTransport` evaluates to `true` when the request uses HTTPS and to `false` when it uses HTTP. An explicit `Deny` statement conditioned on `Bool: {"aws:SecureTransport": "false"}` therefore prevents any insecure HTTP request, regardless of permissions that might otherwise be granted through identity-based policies or bucket-policy allow statements. Requests made over HTTPS are not affected by this deny condition and can proceed if the caller has the required S3 and AWS KMS permissions. This policy control protects data in transit independently of the selected server-side encryption method, while the AWS KMS key provides encryption at rest. AWS documents this exact bucket-policy pattern as a way to require SSL/TLS for Amazon S3 requests. See Amazon S3 policy keys and the aws:SecureTransport global condition key.

QUESTION NO: 48

A developer is using AWS CodeBuild as part of a continuous integration pipeline. The build process needs to retrieve a database password from AWS Secrets Manager and use the password during integration tests. The developer must avoid storing the password in the buildspec file or in source code.

Which actions should the developer take? (Choose two.)

A. Grant the CodeBuild service role permission to call secretsmanager:GetSecretValue for the required secret.

B. Configure the buildspec file to map the Secrets Manager secret value to an environment variable.

C. Add the database password as a plaintext environment variable in the CodeBuild project configuration.

D. Store the database password in an encrypted Amazon S3 object and download the object during every build.

E. Attach the secret directly to the CodeBuild build artifact.

ANSWER: A B

Explanation:

Grant the CodeBuild service role permission to call `secretsmanager:GetSecretValue` for the required secret. CodeBuild uses its service role credentials when it accesses AWS services during a build. The policy should be scoped to the specific secret ARN whenever possible.

Configure the buildspec file to map the Secrets Manager secret value to an environment variable. CodeBuild supports retrieving a secret value from Secrets Manager as an environment variable. The secret reference remains in the build configuration instead of storing the plaintext password in source code or the buildspec file. See CodeBuild buildspec reference for Secrets Manager and Retrieve secrets from AWS Secrets Manager.

QUESTION NO: 49

A developer is building an application that uses an Amazon RDS for PostgreSQL database. To meet security requirements, the developer needs to ensure that data is encrypted at rest. The developer must be able to rotate the encryption keys on demand.

A. Use an AWS KMS managed encryption key to encrypt the database.

B. Create a symmetric customer managed AWS KMS key. Use the key to encrypt the database.

C. Create a 256-bit AES-GCM encryption key. Store the key in AWS Secrets Manager, and enable managed rotation. Use the key to encrypt the database.

D. Create a 256-bit AES-GCM encryption key. Store the key in AWS Secrets Manager. Configure an AWS Lambda function to perform key rotation. Use the key to encrypt the database.

ANSWER: B

Explanation:

Create a symmetric customer managed AWS KMS key. Use the key to encrypt the database. Amazon RDS supports encryption at rest for DB instances by using AWS Key Management Service (AWS KMS) keys, and a customer managed KMS key gives the customer control over the key's policy, permissions, lifecycle, and rotation configuration. RDS uses envelope encryption: it uses the KMS key to protect data-encryption keys that encrypt the underlying database storage, automated backups, read replicas, and snapshots. This lets the developer meet the encryption-at-rest requirement without implementing database-level key storage or cryptographic operations in the application.

A customer managed symmetric KMS key is the appropriate key type for Amazon RDS storage encryption. AWS KMS supports rotation for eligible customer managed keys, including configurable automatic rotation periods, and key rotation is handled transparently by KMS so existing encrypted data remains accessible. The customer can also control when to update rotation settings and can use manual key-material rotation where that model applies. Using a customer managed key therefore provides the required operational control while retaining the managed integration between RDS and KMS. See Encrypting Amazon RDS resources and Rotating AWS KMS keys.

QUESTION NO: 50

A company is using an AWS Lambda function to process records from an Amazon Kinesis data stream. The company recently observed slow processing of the records. A developer notices that the iterator age metric for the function is increasing and that the Lambda run duration is constantly above normal.

Which actions should the developer take to increase the processing speed? (Choose two.)

A. Increase the number of shards of the Kinesis data stream.

B. Decrease the timeout of the Lambda function.

C. Increase the memory that is allocated to the Lambda function.

D. Decrease the number of shards of the Kinesis data stream.

E. Increase the timeout of the Lambda function.

ANSWER: A C

Explanation:

Increasing the number of shards of the Kinesis data stream increases the stream's available read capacity and the potential concurrent processing available to a Lambda event source mapping. Lambda processes records from Kinesis with ordering preserved within each shard, so each shard provides a distinct unit that can be processed concurrently. When iterator age is rising, records are waiting longer before processing; adding appropriately distributed shards can help the consumer keep pace with the incoming record volume and reduce that backlog.

Increasing the memory allocated to the Lambda function can reduce abnormally long invocation durations. AWS Lambda allocates CPU capacity proportionally to the configured memory setting, along with other resources. Therefore, a function whose record-processing work is CPU-bound, or benefits from additional resources, can execute more quickly after a memory increase. Faster processing per invocation raises the rate at which each shard's records are handled, helping reduce iterator age. The developer should validate the new settings with metrics and load testing, while ensuring records are partitioned across the added Kinesis shards. See the AWS Lambda documentation for using Kinesis as an event source and Lambda memory and CPU configuration guidance.

QUESTION NO: 51

A developer maintains applications that store several secrets in AWS Secrets Manager. The applications use secrets that have changed over time. The developer needs to identify required secrets that are still in use. The developer does not want to cause any application downtime.

What should the developer do to meet these requirements?

A. Configure an AWS CloudTrail log file delivery to an Amazon S3 bucket. Create an Amazon CloudWatch alarm for the GetSecretValue. Secrets Manager API operation requests

B. Create a secrets manager-secret-unused AWS Config managed rule. Create an Amazon EventBridge rule to Initiate notification when the AWS Config managed rule is met.

C. Deactivate the applications secrets and monitor the applications error logs temporarily.

D. Configure AWS X-Ray for the applications. Create a sampling rule lo match the GetSecretValue Secrets Manager API operation requests.

ANSWER: B

Explanation:

Create a secrets manager-secret-unused AWS Config managed rule. Create an Amazon EventBridge rule to Initiate notification when the AWS Config managed rule is met. This approach identifies potentially unused Secrets Manager secrets by evaluating their access history without changing, disabling, or deleting any secret. The AWS Config managed rule evaluates whether a secret has been accessed within a configurable number of days. This enables the developer to distinguish secrets that remain actively required by applications from legacy secrets that are no longer being retrieved. Because the evaluation is observational, applications continue to retrieve their current secrets normally, so the process does not introduce downtime.

AWS Config can publish rule compliance-change events to Amazon EventBridge. An EventBridge rule can then route noncompliance findings to a notification target, such as an Amazon SNS topic, allowing the developer to review candidates

before making any cleanup decision. This provides a controlled and auditable process: first identify secrets with no recent access, then notify the appropriate team, validate ownership and dependencies, and only afterward remove or retire secrets if appropriate. The managed rule and event-driven notification together meet the need to find secrets still in use while avoiding disruptive testing in production. See the AWS Config secretsmanager-secret-unused managed rule documentation and AWS Config events in Amazon EventBridge.

QUESTION NO: 52

A company is using Amazon API Gateway to invoke a new AWS Lambda function The company has Lambda function versions in its PROD and DEV environments. In each environment, there is a Lambda function alias pointing to the corresponding Lambda function version API Gateway has one stage that is configured to point at the PROD alias

The company wants to configure API Gateway to enable the PROD and DEV Lambda function versions to be simultaneously and distinctly available

Which solution will meet these requirements?

A. Enable a Lambda authorizer for the Lambda function alias in API Gateway Republish PROD and create a new stage for DEV Create API Gateway stage variables for the PROD and DEV stages. Point each stage variable to the PROD Lambda authorizer to the DEV Lambda authorizer.

B. Set up a gateway response in API Gateway for the Lambda function alias. Republish PROD and create a new stage for DEV. Create gateway responses in API Gateway for PROD and DEV Lambda aliases

C. Use an environment variable for the Lambda function alias in API Gateway. Republish PROD and create a new stage for development. Create API gateway environment variables for PROD and DEV stages. Point each stage variable to the PROD Lambda function alias to the DEV Lambda function alias.

D. Use an API Gateway stage variable to configure the Lambda function alias Republish PROD and create a new stage for development Create API Gateway stage variables for PROD and DEV stages Point each stage variable to the PROD Lambda function alias and to the DEV Lambda function alias

ANSWER: D

Explanation:

Use an API Gateway stage variable to configure the Lambda function alias Republish PROD and create a new stage for development Create API Gateway stage variables for PROD and DEV stages Point each stage variable to the PROD Lambda function alias and to the DEV Lambda function alias. API Gateway stages provide separate, addressable deployment environments, so creating distinct PROD and development stages lets clients invoke each environment independently through its stage-specific URL. A stage variable can hold the Lambda alias name and be referenced in the integration URI, for example by using `${stageVariables.lambdaAlias}` as the alias qualifier in the Lambda invocation ARN. The PROD stage variable resolves to the PROD alias, while the development stage variable resolves to the DEV alias. Consequently, the same API configuration and resource paths can route requests to distinct, version-qualified Lambda aliases depending on the selected stage. This approach supports independent deployments and configuration changes without requiring separate APIs. API Gateway also needs permission to invoke each qualified Lambda alias; Lambda resource-based permissions can be scoped to the API Gateway execution ARN and alias. AWS documents using stage variables in Lambda integrations and identifies stages as logical references to deployed API configurations. See API Gateway stage variables and Lambda function aliases.

QUESTION NO: 53

A developer is configuring an applications deployment environment in AWS CodePipeine. The application code is stored in a GitHub repository. The developer wants to ensure that the repository package ' s unit tests run in the new deployment environment. The deployment has already set the pipeline ' s source provider to GitHub and has specified the repository and branch to use in the deployment.

When combination of steps should the developer take next to meet these requirements with the least the LEAST overhead ' (Select TWO).

A. Create an AWS CodeCommit project. Add the repository package's build and test commands to the project's buildspec.

B. Create an AWS CodeBuild project. Add the repository package's build and test commands to the project's buildspec.

C. Create an AWS CodeDeploy project. Add the repository package's build and test commands to the project's buildspec.

D. Add an action to the source stage. Specify the newly created project as the action provider. Specify the build artifact as the action's input artifact.

E. Add a new stage to the pipeline after the source stage. Add an action to the new stage. Specify the newly created project as the action provider. Specify the source artifact as the action's input artifact.

ANSWER: B E

Explanation:

Creating an AWS CodeBuild project and placing the package build and unit-test commands in its buildspec provides a managed, purpose-built execution environment for compiling, testing, and packaging application code. CodeBuild can obtain the source artifact passed by CodePipeline, execute the commands defined in `buildspec.yml`, and report the build result to the pipeline. The buildspec can define phases such as `install`, `pre_build`, `build`, and `post_build`, allowing unit tests to run as part of the automated build workflow without operating test servers or custom orchestration infrastructure.

Adding a stage after the GitHub source stage and configuring a build action that uses the new CodeBuild project connects the source artifact to that test process. CodePipeline passes the output artifact from the source action as the input artifact to the CodeBuild action. The pipeline can then proceed to deployment only after the build action completes successfully, so failed unit tests stop the release before the application is deployed. This arrangement uses fully managed CodePipeline and CodeBuild services, which minimizes operational overhead while keeping source retrieval, testing, and deployment clearly separated. See the AWS CodePipeline CodeBuild action reference and the AWS CodeBuild buildspec reference.

QUESTION NO: 54

A developer has an application that makes batch requests directly to Amazon DynamoDB by using the BatchGetItem low-level API operation. The responses frequently return values in the UnprocessedKeys element.

Which actions should the developer take to increase the resiliency of the application when the batch response includes values in UnprocessedKeys? (Choose two.)

A. Retry the batch operation immediately.

B. Retry the batch operation with exponential backoff and randomized delay.

C. Update the application to use an AWS software development kit (AWS SDK) to make the requests.

D. Increase the provisioned read capacity of the DynamoDB tables that the operation accesses.

E. Increase the provisioned write capacity of the DynamoDB tables that the operation accesses.

ANSWER: B D

Explanation:

Retry the batch operation with exponential backoff and randomized delay. is appropriate because `UnprocessedKeys` identifies requested items that DynamoDB did not process during that particular `BatchGetItem` call. DynamoDB explicitly recommends retrying those keys with an exponential backoff algorithm. Adding randomized delay (jitter) spreads retries across time when multiple application workers experience throttling simultaneously, reducing contention and allowing DynamoDB capacity to recover. The retry should submit only the keys returned in `UnprocessedKeys`, not blindly repeat all keys from the original batch request.

Increase the provisioned read capacity of the DynamoDB tables that the operation accesses. is also correct when frequent unprocessed keys result from insufficient read throughput. `BatchGetItem` consumes read capacity units, and DynamoDB can return unprocessed keys if processing the full request would exceed provisioned throughput. Increasing the available read capacity gives the workload more throughput headroom and reduces the frequency of throttling-driven partial batch results. Together, adequate read capacity and resilient, backoff-based retries provide both a preventive capacity

measure and reliable handling of transient partial processing. See the BatchGetItem API reference and DynamoDB batch operation error handling guidance.

QUESTION NO: 55

A developer is designing an event-driven architecture. An AWS Lambda function that processes data needs to push processed data to a subset of four consumer Lambda functions. The data must be routed based on the value of one field in the data.

Which solution will meet these requirements with the LEAST operational overhead?

A. Create an Amazon Simple Queue Service {Amazon SQS) queue and even! source mapping for each consumer Lambda function. Add message routing logic to the data-processing Lambda function.

B. Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the four consumer Lambda functions to the topic. Add message filtering logic to each consumer Lambda function. Subscribe the data-processing Lambda function to the SNS topic.

C. Create a separate Amazon Simple Notification Service (Amazon SNS) topic and subscription for each consumer Lambda function. Add message routing logic to the data-processing Lambda function to publish to the appropriate topic.

D. Create a single Amazon Simple Notification Service {Amazon SNS) topic. Subscribe the four consumer Lambda functions to the topic. Add SNS subscription filter policies to each subscription. Configure the data-processing Lambda function to publish to the topic.

ANSWER: D

Explanation:

Create a single Amazon Simple Notification Service {Amazon SNS) topic. Subscribe the four consumer Lambda functions to the topic. Add SNS subscription filter policies to each subscription. Configure the data-processing Lambda function to publish to the topic.

Amazon SNS subscription filter policies provide managed, declarative message routing at the topic subscription level. The data-processing Lambda function publishes each processed event once to one SNS topic, including the routing field in the message body or message attributes. Each Lambda subscription then has a filter policy that specifies the field values for which that consumer should receive events. SNS evaluates the policy before invoking the subscribed Lambda function, so only the applicable subset of consumer functions receives each event.

This design keeps routing logic out of the publisher and avoids requiring every consumer function to receive, inspect, and discard irrelevant messages. It also centralizes the fan-out and filtering configuration in SNS, making it straightforward to add, update, or remove consumers without changing the data-processing Lambda function. SNS supports filter policies based on message attributes and, when configured with a message body filter scope, JSON message body properties. See the Amazon SNS subscription filter policy documentation and SNS delivery to AWS Lambda documentation.

QUESTION NO: 56

An application interacts with Amazon Aurora to store and track customer information. The primary database is set up with multiple read replicas for improving the performance of the read queries. However, one of the Aurora replicas is receiving most or all of the traffic, while the other Aurora replica remains idle.

How can this issue be resolved?

A. Disable application-level DNS caching.

B. Enable application-level DNS caching.

C. Enable application pooling.

D. Disable application pooling.

ANSWER: A

Explanation:

Disable application-level DNS caching. Aurora's reader endpoint is designed to distribute read-only connections across the available Aurora Replica instances. It does this through DNS resolution: the reader endpoint returns replica IP addresses so that newly established client connections can be directed to different readers. If the application caches the DNS result itself for too long, it can continue connecting to the same resolved replica address rather than performing fresh DNS lookups. As a result, one replica can receive nearly all read traffic while another has little or no activity.

Disabling application-managed DNS caching allows the application to honor the Aurora endpoint's DNS-based connection distribution behavior. New connections can then resolve the reader endpoint again and be distributed among the replicas. The application should use the Aurora reader endpoint, rather than an individual replica instance endpoint, for read workloads. This behavior is especially relevant when the application creates new database connections over time; existing long-lived connections remain attached to the replica selected when each connection was established.

AWS documents that the Aurora reader endpoint provides load balancing for read-only connections and that DNS caching can interfere with this distribution. See the Amazon Aurora endpoints documentation and Aurora Replicas documentation.

QUESTION NO: 57

A developer is deploying an application on Amazon EC2 instances that run in Account A. In certain cases, this application needs to read data from a private Amazon S3 bucket in Account B. The developer must provide the application access to the S3 bucket without exposing the S3 bucket to anyone else.

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

A. Create an IAM role with S3 read permissions in Account B.

B. Update the instance profile IAM role in Account A with S3 read permissions.

C. Make the S3 bucket public with limited access for Account A.

D. Configure the bucket policy in Account B to grant permissions to the instance profile role.

E. Add a trust policy that allows s3:Get* permissions to the IAM role in Account B.

ANSWER: B D

Explanation:

Cross-account access to an Amazon S3 bucket is authorized through permissions from both sides of the access relationship. The EC2 instance profile role in Account A needs an identity-based IAM policy that permits the required read actions, such as `s3:GetObject` and, where the application needs to enumerate keys, `s3:ListBucket`. The application obtains its temporary credentials automatically from this instance profile role, so it does not need embedded or shared long-term credentials.

The S3 bucket owner in Account B must also explicitly delegate access with a bucket policy. The policy should specify the ARN of the Account A instance profile role as its `Principal` and limit the allowed actions and resources to the required bucket and object paths. This resource-based authorization keeps the bucket private: only sessions using the specified role can read the permitted data. The bucket policy can further restrict access to a prefix or use conditions when appropriate.

AWS documents this model as cross-account access using an IAM role and an S3 bucket policy. See Amazon S3 bucket policy examples and IAM cross-account resource access.

QUESTION NO: 58

A company has an application that runs on Amazon EC2 instances and stores sensitive data. The application encrypts the data by using an AWS KMS customer managed key with key material that has been imported from outside of AWS.

A developer accidentally deletes the key material, and the application is unable to read the encrypted data.

What should the developer do to fix the application with the LEAST amount of effort?

A. Create new key material from the original source outside of AWS. Reimport the key material to the existing KMS key.

B. Recover the deleted key material by opening a request with AWS Support.

C. Generate an AWS managed KMS key. Reconfigure the application to use this key.

D. Reimport the key material from the original source outside of AWS to the existing KMS key.

ANSWER: D

Explanation:

Reimport the key material from the original source outside of AWS to the existing KMS key. Imported key material is cryptographic material that the customer generated externally and imported into a customer managed KMS key. If that material is deleted, AWS KMS retains the KMS key object and its metadata, but the key cannot perform cryptographic operations while it has no key material. This makes data encrypted under that KMS key unavailable until the required material is restored.

To decrypt the existing ciphertext, the developer must reimport the same original cryptographic key material into the same KMS key. Restoring it to that key preserves the key identity and cryptographic capability required to decrypt data that was previously encrypted. The import process uses current KMS import parameters, including a wrapping key and import token, but the externally sourced key material itself must be the original material used by the KMS key. This is the least-effort recovery because it restores access without changing the application or re-encrypting existing data. AWS documents that deleted imported key material can be reimported, provided the original material remains available. See Import key material into AWS KMS keys and Deleting imported key material.

QUESTION NO: 59

A company is building a micro services app1 cation that consists of many AWS Lambda functions. The development team wants to use AWS Serverless Application Model (AWS SAM) templates to automatically test the Lambda functions. The development team plans to test a small percentage of traffic that is directed to new updates before the team commits to a full deployment of the application.

Which combination of steps will meet these requirements in the MOST operationally efficient way? (Select TWO.)

A. Use AWS SAM CLI commands in AWS CodeDeploy lo invoke the Lambda functions to lest the deployment

B. Declare the EventInvokeConfig on the Lambda functions in the AWS SAM templates with OnSuccess and OnFailure configurations.

C. Enable gradual deployments through AWS SAM templates.

D. Set the deployment preference type to Canary10Percent30Minutes. Use hooks to test the deployment.

E. Set the deployment preference type to Linear10PefcentEvery10Minutes Use hooks to test the deployment.

ANSWER: C D

Explanation:

Enable gradual deployments through AWS SAM templates because AWS SAM provides built-in support for safe, incremental AWS Lambda deployments. A function's `DeploymentPreference` configuration instructs SAM to create and manage the AWS CodeDeploy deployment resources needed to publish a new Lambda version, route traffic through an alias, monitor the deployment, and roll back when alarms indicate a problem. This removes the need for the team to build custom traffic-shifting automation.

Set the deployment preference type to `Canary10Percent30Minutes` and use hooks to test the deployment because this predefined canary configuration directs 10% of invocations to the updated Lambda version for 30 minutes before CodeDeploy shifts the remaining traffic. SAM deployment hooks provide `PreTraffic` and `PostTraffic` validation Lambda functions. The validation functions can run automated checks as part of the deployment lifecycle and report success

or failure to CodeDeploy. This combines controlled exposure of the new code with automated deployment validation, allowing the application to proceed to full traffic only after the canary period and validation complete successfully. See the AWS SAM DeploymentPreference documentation and the AWS CodeDeploy Lambda deployment configuration documentation.

QUESTION NO: 60

A developer is building an application that needs to access the values of secrets that are in AWS Secrets Manager. The secret IDs are passed to the application code through environment variables. The secrets are encrypted by a customer managed AWS KMS key.

Which combination of permissions is required to retrieve the values of these secrets? (Select TWO.)

A. secretsmanager:GetSecretValue

B. secretsmanager:DescribeSecret

C. secretsmanager:ListSecrets

D. kms:Decrypt

E. kms:Encrypt

ANSWER: A D

Explanation:

Retrieving the plaintext value of an AWS Secrets Manager secret requires `secretsmanager:GetSecretValue` for the target secret. This is the Secrets Manager API action that returns the secret's `SecretString` or `SecretBinary`. Supplying a secret ID through an environment variable merely provides an identifier that the application can use in its request; the application's IAM principal still requires authorization to read the secret value.

Because these secrets use a customer managed AWS KMS key, the calling principal also needs `kms:Decrypt` permission for that KMS key. Secrets Manager uses the configured KMS key to decrypt the protected secret data when it services a `GetSecretValue` request. The KMS key policy must also permit this use, either by enabling IAM policies or by granting access directly through the key policy or a KMS grant. AWS documents that applications retrieving secrets encrypted under a customer managed key require KMS decrypt access in addition to permission to retrieve the secret.

Therefore, the necessary permissions are `secretsmanager:GetSecretValue` on the secret and `kms:Decrypt` on the customer managed KMS key. See the AWS Secrets Manager identity-based policy documentation and the AWS Secrets Manager encryption documentation.

QUESTION NO: 61

A developer has built an application that inserts data into an Amazon DynamoDB table. The table is configured to use provisioned capacity. The application is deployed on a burstable nano Amazon EC2 instance. The application logs show that the application has been failing because of a ProvisionedThroughputExceededException error.

Which actions should the developer take to resolve this issue? (Select TWO.)

A. Move the application to a larger EC2 instance.

B. Increase the number of read capacity units (RCUs) that are provisioned for the DynamoDB table.

C. Reduce the frequency of requests to DynamoDB by implementing exponential backoff.

D. Increase the frequency of requests to DynamoDB by decreasing the retry delay.

E. Change the capacity mode of the DynamoDB table from provisioned to on-demand.

ANSWER: C E

Explanation:

“Reduce the frequency of requests to DynamoDB by implementing exponential backoff” is correct because DynamoDB recommends retrying throttled requests with exponential backoff. Backoff progressively increases the delay between retries, reducing contention and allowing the table’s available capacity to recover. In production applications, adding jitter to the backoff interval is also a common practice to prevent many clients from retrying at the same moment. This makes transient throttling less likely to result in repeated failures.

“Change the capacity mode of the DynamoDB table from provisioned to on-demand” is also correct. The workload is performing inserts, which consume write capacity. On-demand capacity mode automatically accommodates changing request volumes without the developer having to calculate and manually provision write capacity units. This directly addresses throttling caused by a write workload exceeding the table’s configured provisioned throughput. DynamoDB can switch between provisioned and on-demand capacity modes, subject to service rules for mode changes. See the AWS guidance for DynamoDB error handling and exponential backoff and DynamoDB read/write capacity modes.

QUESTION NO: 62

A company deploys a new application to AWS. The company is streaming application logs to Amazon CloudWatch Logs. The company ' s development team must receive notification by email when the word " ERROR " appears in any log lines. A developer sets up an Amazon SNS topic and subscribes the development team to the topic.

What should the developer do next to meet the requirements?

A. Select the appropriate log group. Create a CloudWatch metric filter with " ERROR " as the search term. Create an alarm on this metric that notifies the SNS topic when the metric is 1 or higher.

B. In CloudWatch Logs Insights, select the appropriate log group. Create a metric query to search for the term " ERROR " in the logs. Create an alarm on this metric that notifies the SNS topic when the metric is 1 or higher.

C. Select the appropriate log group. Create an SNS subscription filter with " ERROR " as the filter pattern. Select the SNS topic as the destination.

D. Create a CloudWatch alarm that includes " ERROR " as a filter pattern, a log group dimension that defines the appropriate log group, and a destination that notifies the SNS topic.

ANSWER: A

Explanation:

Select the appropriate log group. Create a CloudWatch metric filter with " ERROR " as the search term. Create an alarm on this metric that notifies the SNS topic when the metric is 1 or higher. This creates the required automated detection-and-notification workflow. A CloudWatch Logs metric filter evaluates incoming log events in the selected log group against the specified filter pattern. Each event containing `ERROR` causes the filter to publish a metric data point through its metric transformation. A CloudWatch alarm can then evaluate that custom metric over a chosen period and enter the ALARM state when at least one matching event is detected. The alarm action publishes a notification to the existing SNS topic, and SNS delivers the message to the subscribed email recipients. The alarm should use an appropriate period and threshold of 1 or higher so a single matching log line is sufficient to trigger notification. This approach works continuously as new logs arrive and does not require a person to run queries or inspect logs manually. For implementation details, see the AWS documentation for monitoring log data with metric filters and creating an alarm that sends an SNS notification.