Amazon.DVA-C02.v2026-07-03.q251

Exam Code:	DVA-C02
Exam Name:	AWS Certified Developer - Associate
Certification Provider:	Amazon
Free Question Number:	251
Version:	v2026-07-03
# of views:	313
# of Questions views:	2510
https://www.freepdfdumps.com/Amazon.DVA-C02.v2026-07-03.q251.html	

NEW QUESTION: 1
A company uses Amazon API Gateway to expose a set of APIs to customers. The APIs have caching enabled in API Gateway. Customers need a way to invalidate the cache for each API when they test the API.
What should a developer do to give customers the ability to invalidate the API cache?
A. Ask the customers to use the AWS SDK API Gateway class to invoke the InvalidateCache API operation.
B. Ask the customers to use AWS credentials to call the InvalidateCache API operation.
C. Attach an InvalidateCache policy to the IAM execution role that the customers use to invoke the API.Ask the customers to add the INVALIDATE_CACHE query string parameter when they make an API call.
D. Attach an InvalidateCache policy to the IAM execution role that the customers use to invoke the API.
Ask the customers to send a request that contains the HTTP header when they make an API call.
Answer: (SHOW ANSWER)

NEW QUESTION: 2
A developer needs to use a code template to create an automated deployment of an application onto Amazon EC2 instances. The template must be configured to repeat deployment, installation, and updates of resources for the application. The template must be able to create identical environments and roll back to previous versions.
Which solution will meet these requirements?
A. Use AWS Amplify for automatic deployment templates. Use a traffic-splitting deployment to copy any deployments. Modify any resources created by Amplify, if necessary.

B. Use AWS AppSync to deploy the application. Upload the template as a GraphQL schema. Specify the EC2 instances for deployment of the application. Use resolvers as a version control mechanism and to make any updates to the deployments.
C. Use AWS CodeBuild for automatic deployment. Upload the required AppSpec file template. Save the appspec.yml file in the root directory folder of the revision. Specify the deployment group that includes the EC2 instances for the deployment.
D. Use AWS CloudFormation to create an infrastructure template in JSON format to deploy the EC2 instances. Use Cloud Formation helper scripts to install the necessary software and to start the application. Call the scripts directly from the template.
Answer: D (LEAVE A REPLY)

NEW QUESTION: 3
A developer needs to retrieve all data from an Amazon DynamoDB table that matches a particular partition key.
Which solutions will meet this requirement in the MOST operationally efficient way? (Select TWO.)
A. Use the GetItem API and a PartiQL statement to match on the key.
B. Use the Scan API and a filter expression to match on the key.
C. Use the GetItem API with a request parameter for key that contains the partition key name and specific key value.
D. Use the ExecuteStatement API and a PartiQL statement to match on the key.
E. Use the ExecuteStatement API and a filter expression to match on the key.
Answer: (SHOW ANSWER)

NEW QUESTION: 4
A company wants to use AWS AppConfig to gradually deploy a new feature to 15% of users to test the feature before a full deployment.
Which solution will meet this requirement with the LEAST operational overhead?
A. Set up a custom script within the application to randomly select 15% of users. Assign a flag for the new feature to the selected users.
B. Create separate AWS AppConfig feature flags for both groups of users. Configure the flags to target
15% of users.
C. Create an AWS AppConfig feature flag. Define a variant for the new feature, and create a rule to target
15% of users.
D. Use AWS AppConfig to create a feature flag without variants. Implement a custom traffic-splitting mechanism in the application code.
Answer: (SHOW ANSWER)
AWS AppConfig Feature Flags are designed to release features safely with minimal code changes. The lowest operational overhead approach is to use a single feature flag and

configure it to deliver the feature to a percentage of users through built-in targeting rules and variants.
With option C, the developer creates one AppConfig feature flag and defines a variant that represents the new feature behavior (for example, enabled=true or a "newExperience" variant). Then the developer creates a rule in AppConfig to target 15% of users. AppConfig feature flags support rules that can evaluate attributes (such as user IDs or other contextual attributes supplied by the application) and apply percentage-based rollout. This avoids building and maintaining custom rollout logic in the application.
Option A and D both require implementing and maintaining custom traffic splitting in code, which increases complexity and operational burden, and risks inconsistent behavior across clients/services.
Option B adds extra configuration overhead and complexity by managing multiple feature flags for "groups" instead of using a single flag with a variant and rule. A single flag with variants is the clean, scalable pattern.
Therefore, create one AppConfig feature flag, define a variant, and configure a rule to target 15% of users.

NEW QUESTION: 5
A developer wants to store information about movies. Each movie has a title, release year, and genre. The movie information also can include additional properties about the cast and production crew. This additional information is inconsistent across movies. For example, one movie might have an assistant director, and another movie might have an animal trainer.
The developer needs to implement a solution to support the following use cases:
For a given title and release year, get all details about the movie that has that title and release year.
For a given title, get all details about all movies that have that title.
For a given genre, get all details about all movies in that genre.
Which data store configuration will meet these requirements?
A. Create an Amazon DynamoDB table. Configure the table with a primary key that consists of the title as the partition key and the release year as the sort key. Create a global secondary index that uses the genre as the partition key and the title as the sort key.
B. Create an Amazon DynamoDB table. Configure the table with a primary key that consists of the genre as the partition key and the release year as the sort key. Create a global secondary index that uses the title as the partition key.
C. On an Amazon RDS DB instance, create a table that contains columns for title, release year, and genre.
Configure the title as the primary key.
D. On an Amazon RDS DB instance, create a table where the primary key is the title and all other data is encoded into JSON format as one additional column.
Answer: (SHOW ANSWER)

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and consistent performance with seamless scalability. The developer can create a DynamoDB table and configure the table with a primary key that consists of the title as the partition key and the release year as the sort key. This will enable querying for a given title and release year efficiently. The developer can also create a global secondary index that uses the genre as the partition key and the title as the sort key. This will enable querying for a given genre efficiently. The developer can store additional properties about the cast and production crew as attributes in the DynamoDB table. These attributes can have different data types and structures, and they do not need to be consistent across items.
References:
* [Amazon DynamoDB]
* [Working with Queries - Amazon DynamoDB]
* [Working with Global Secondary Indexes - Amazon DynamoDB]

NEW QUESTION: 6
A development team wants to build a continuous integration/continuous delivery (CI/CD) pipeline. The team is using AWS CodePipeline to automate the code build and deployment. The team wants to store the program code to prepare for the CI/CD pipeline. Which AWS service should the team use to store the program code?
A. AWS CodeDeploy
B. AWS CodeArtifact
C. AWS CodeCommit
D. Amazon CodeGuru
Answer: (SHOW ANSWER)
AWS CodeCommit is a service that provides fully managed source control for hosting secure and scalable private Git repositories. The development team can use CodeCommit to store the program code and prepare for the CI/CD pipeline. CodeCommit integrates with other AWS services such as CodePipeline, CodeBuild, and CodeDeploy to automate the code build and deployment process.
References:
* [What Is AWS CodeCommit? - AWS CodeCommit]
* [AWS CodePipeline - AWS CodeCommit]

NEW QUESTION: 7
A company runs an AWS CodeBuild project on medium-sized Amazon EC2 instances. The company wants to cost optimize the project and reduce the provisioning time.
A. Configure the project to run on a CodeBuild reserved capacity fleet.
B. Select AWS Lambda as the compute mode for the CodeBuild project.
C. Configure the project to run on a CodeBuild on-demand fleet.
D. Set up Amazon S3 caching for the CodeBuild project.
Answer: D (LEAVE A REPLY)

Comprehensive and Detailed Step-by-Step Explanation:
* Option D: Set up Amazon S3 Caching for CodeBuild:
* CodeBuild supports S3 caching to store intermediate build artifacts and dependencies. This reduces the time required to download dependencies during subsequent builds, effectively lowering costs and improving build performance.
* By using S3 caching, developers can optimize costs without changing the compute type or adding complexity.
* Why Other Options Are Incorrect:
* Option A:CodeBuild does not have a "reserved capacity fleet" option.
* Option B:AWS Lambda cannot be used as the compute mode for CodeBuild projects. CodeBuild uses its own managed build environments.
* Option C:CodeBuild already operates on an on-demand basis, so this does not address the need for optimization or reduced provisioning time.
References:
* AWS CodeBuild Caching Documentation

NEW QUESTION: 8
A company with multiple branch locations has an analytics and reporting application. Each branch office pushes a sales report to a shared Amazon S3 bucket at a predefined time each day. The company has developed an AWS Lambda function that analyzes the reports from all branch offices in a single pass. The Lambda function stores the results in a database.
The company needs to start the analysis once each day at a specific time.
Which solution will meet these requirements MOST cost-effectively?
A. Configure an S3 event notification to invoke the Lambda function when a branch office uploads a sales report.
B. Create an AWS Step Functions state machine that invokes the Lambda function once each day at the predefined time.
C. Configure the Lambda function to run continuously and to begin analysis only at the predefined time each day.
D. Create an Amazon EventBridge scheduled rule that invokes the Lambda function once each day at the predefined time.
Answer: D (LEAVE A REPLY)
The requirement is to run a single analysis job once per day at a specific time, after branch offices have uploaded their daily reports. The most cost-effective and straightforward way to trigger a Lambda function on a fixed schedule is to use Amazon EventBridge scheduled rules (cron or rate expressions). With a scheduled rule, EventBridge invokes the Lambda function at the exact predefined time each day without requiring any continuously running resources, polling logic, or additional orchestration unless it is needed.
Option D fits perfectly: an EventBridge schedule is purpose-built for time-based triggers, and you pay only for the invocations and any EventBridge rule evaluations according to

AWS pricing, with minimal operational overhead. The Lambda function will run exactly once each day and can then read all relevant objects from the S3 bucket and process them in a single pass as designed.
Option A (S3 event notifications) triggers the Lambda function per upload, which is not aligned with "single pass once per day." It could cause multiple invocations and additional cost and complexity (coordination, waiting for all branches, handling partial arrival). Option B (Step Functions) can schedule via EventBridge as well, but Step Functions introduces additional state machine costs and is unnecessary if the requirement is simply a scheduled single Lambda invocation. Option C is the least cost-effective: running continuously to "wait" until a time wastes compute time and increases cost, and it is not an event-driven design.
Therefore, D is the most cost-effective solution: use an EventBridge scheduled rule to invoke the Lambda function once daily at the predefined time.

NEW QUESTION: 9
A developer wants the ability to roll back to a previous version of an AWS Lambda function in the event of errors caused by a new deployment. How can the developer achieve this with MINIMAL impact on users?
A. Change the application to use an alias that points to the current version. Deploy the new version of the code Update the alias to use the newly deployed version. If too many errors are encountered, point the alias back to the previous version.
B. Change the application to use an alias that points to the current version. Deploy the new version of the code. Update the alias to direct 10% of users to the newly deployed version. If too many errors are encountered, send 100% of traffic to the previous version
C. Do not make any changes to the application. Deploy the new version of the code. If too many errors are encountered, point the application back to the previous version using the version number in the Amazon Resource Name (ARN).
D. Create three aliases: new, existing, and router. Point the existing alias to the current version. Have the router alias direct 100% of users to the existing alias. Update the application to use the router alias.
Deploy the new version of the code. Point the new alias to this version. Update the router alias to direct
10% of users to the new alias. If too many errors are encountered, send 100% of traffic to the existing alias.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 10
An application uses Lambda functions to extract metadata from files uploaded to an S3 bucket; the metadata is stored in Amazon DynamoDB. The application starts behaving unexpectedly, and the developer wants to examine the logs of the Lambda function code for errors.

Based on this system configuration, where would the developer find the logs?

A. Amazon S3

B. AWS CloudTrail

C. Amazon CloudWatch

D. Amazon DynamoDB

Answer: C (LEAVE A REPLY)

Amazon CloudWatch is the service that collects and stores logs from AWS Lambda functions. The developer can use CloudWatch Logs Insights to query and analyze the logs for errors and metrics. Option A is not correct because Amazon S3 is a storage service that does not store Lambda function logs. Option B is not correct because AWS CloudTrail is a service that records API calls and events for AWS services, not Lambda function logs. Option D is not correct because Amazon DynamoDB is a database service that does not store Lambda function logs.

References: AWS Lambda Monitoring, [CloudWatch Logs Insights]

NEW QUESTION: 11

Users of a web-based music application are experiencing latency issues on one of the application's most popular pages. A developer identifies that the issue is caused by the slow load time of specific widgets that rank and sort various songs and albums.

The developer needs to ensure that the widgets load more quickly by using built-in, in-memory ranking and sorting techniques. The developer must ensure that the data remains up to date.

Which solution will meet these requirements with the LEAST latency?

A. Provision an Amazon ElastiCache (Memcached) cluster. Implement a lazy-loading caching strategy.

B. Provision an Amazon ElastiCache (Redis OSS) cluster. Implement a write-through caching strategy.

C. Provision an Amazon ElastiCache (Memcached) cluster. Implement a write-through caching strategy.

D. Provision an Amazon ElastiCache (Redis OSS) cluster. Implement a lazy-loading caching strategy.

Answer: B (LEAVE A REPLY)

Comprehensive and Detailed 250 to 300 words of Explanation (AWS-doc-aligned):

To minimize widget latency for ranking and sorting, the best fit is Amazon ElastiCache for Redis OSS because Redis provides built-in in-memory data structures specifically suited for ranking use cases, such as Sorted Sets (ZSETs). Sorted sets maintain members in order by score, which enables fast retrieval of "top N" items and efficient re-ranking without repeatedly querying the primary database and sorting at the application tier. Memcached, by contrast, is a simpler key-value cache and does not provide comparable native ranking /sorting data structures.

To ensure data remains up to date, a write-through caching strategy is preferred. With write-through, updates are written to the cache at the same time as the system of record, so the cache reflects the latest values immediately after writes. This avoids stale rankings that can occur with lazy-loading (cache-aside), where the cache is only populated or refreshed on reads and can serve older data until expiration or explicit invalidation. Combining Redis's native sorted/ranking structures with write-through updates yields the least latency because the page widgets can fetch pre-ranked/pre-sorted results directly from Redis memory with minimal computation, while still maintaining freshness as new plays, likes, or album changes occur. Operationally, this pattern reduces database load, avoids repeated sorting work in the application, and provides consistently fast reads for a high-traffic page.

NEW QUESTION: 12
An developer is building a serverless application by using the AWS Serverless Application Model (AWS SAM). The developer is currently testing the application in a development environment. When the application is nearly finsihed, the developer will need to set up additional testing and staging environments for a quality assurance team.
The developer wants to use a feature of the AWS SAM to set up deployments to multiple environments.
Which solution will meet these requirements with the LEAST development effort?
A. Add a configuration file in TOML format to group configuration entries to every environment. Add a table for each testing and staging environment. Deploy updates to the environments by using the sam deploy command and the --config-env flag that corresponds to the each environment.
B. Create additional AWS SAM templates for each testing and staging environment. Write a custom shell script that uses the sam deploy command and the --template-file flag to deploy updates to the environments.
C. Create one AWS SAM configuration file that has default parameters. Perform updates to the testing and staging environments by using the -parameter-overrides flag in the AWS SAM CLI and the parameters that the updates will override.
D. Use the existing AWS SAM template. Add additional parameters to configure specific attributes for the serverless function and database table resources that are in each environment. Deploy updates to the testing and staging environments by using the sam deploy command.
Answer: A (LEAVE A REPLY)
The correct answer is A. Add a configuration file in TOML format to group configuration entries to every environment. Add a table for each testing and staging environment. Deploy updates to the environments by using the sam deploy command and the --config-env flag that corresponds to the each environment.
A: Add a configuration file in TOML format to group configuration entries to every environment. Add a table for each testing and staging environment. Deploy updates to the

environments by using the sam deploy command and the --config-env flag that corresponds to the each environment. This is correct. This solution will meet the requirements with the least development effort, because it uses a feature of the AWS SAM CLI that supports a project-level configuration file that can be used to configure AWS SAM CLI command parameter values1. The configuration file can have multiple environments, each with its own set of parameter values, such as stack name, region, capabilities, and more2. The developer can use the --config-env option to specify which environment to use when deploying the application3. This way, the developer can avoid creating multiple templates or scripts, or manually overriding parameters for each environment.
B: Create additional AWS SAM templates for each testing and staging environment. Write a custom shell script that uses the sam deploy command and the --template-file flag to deploy updates to the environments.
This is incorrect. This solution will not meet the requirements with the least development effort, because it requires creating and maintaining multiple templates and scripts for each environment. This can introduce duplication, inconsistency, and complexity in the deployment process.
C: Create one AWS SAM configuration file that has default parameters. Perform updates to the testing and staging environments by using the -parameter-overrides flag in the AWS SAM CLI and the parameters that the updates will override. This is incorrect. This solution will not meet the requirements with the least development effort, because it requires manually specifying and overriding parameters for each environment every time the developer deploys the application. This can be error-prone, tedious, and inefficient.
D: Use the existing AWS SAM template. Add additional parameters to configure specific attributes for the serverless function and database table resources that are in each environment. Deploy updates to the testing and staging environments by using the sam deploy command. This is incorrect. This solution will not meet the requirements with the least development effort, because it requires modifying the existing template and adding complexity to the resource definitions for each environment. This can also make it difficult to manage and track changes across different environments.
References:
* 1: AWS SAM CLI configuration file - AWS Serverless Application Model
* 2: Configuration file basics - AWS Serverless Application Model
* 3: Specify a configuration file - AWS Serverless Application Model

NEW QUESTION: 13
A developer is using AWS CodeDeploy to launch an application onto Amazon EC2 instances. The application deployment fails during testing. The developer notices an IAM_ROLE_PERMISSIONS error code in Amazon CloudWatch logs.
What should the developer do to resolve the error?
A. Ensure the CodeDeploy agent is installed and running on all instances in the deployment group.

B. Attach the AWSCodeDeployRoleECS policy to the CodeDeploy service role.
C. Ensure that the deployment group is using the correct role name for the CodeDeploy service role.
D. Attach the AWSCodeDeployRole policy to the CodeDeploy service role.
Answer: (SHOW ANSWER)

NEW QUESTION: 14
A data visualization company wants to strengthen the security of its core applications The applications are deployed on AWS across its development staging, pre-production, and production environments. The company needs to encrypt all of its stored sensitive credentials The sensitive credentials need to be automatically rotated Aversion of the sensitive credentials need to be stored for each environment Which solution will meet these requirements in the MOST operationally efficient way?
A. Configure AWS Secrets Manager versions to store different copies of the same credentials across multiple environments
B. Create a new parameter version in AWS Systems Manager Parameter Store for each environment Store the environment-specific credentials in the parameter version.
C. Configure the environment variables in the application code Use different names for each environment type
D. Configure AWS Secrets Manager to create a new secret for each environment type. Store the environment-specific credentials in the secret
Answer: D (LEAVE A REPLY)
* Secrets Management: AWS Secrets Manager is designed specifically for storing and managing sensitive credentials.
* Environment Isolation: Creating separate secrets for each environment (development, staging, etc.) ensures clear separation and prevents accidental leaks.
* Automatic Rotation: Secrets Manager provides built-in rotation capabilities, enhancing security posture.
* Versioning: Tracking changes to secrets is essential for auditing and compliance.
References:
* AWS Secrets Manager: https://aws.amazon.com/secrets-manager/
* Secrets Manager Rotation:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating- secrets.html

NEW QUESTION: 15
A company needs to develop a proof of concept for a web service application. The application will show the weather forecast for one of the company's office locations. The application will provide a REST endpoint that clients can call. Where possible, the application should use caching features provided by AWS to limit the number of requests to the backend service. The application backend will receive a small amount of traffic only during testing.

Which approach should the developer take to provide the REST endpoint MOST cost-effectively?

A. Create a container image. Deploy the container image by using Amazon EKS. Expose the functionality by using Amazon API Gateway.

B. Create an AWS Lambda function by using AWS SAM. Expose the Lambda functionality by using Amazon API Gateway.

C. Create a container image. Deploy the container image by using Amazon ECS. Expose the functionality by using Amazon API Gateway.

D. Create a microservices application. Deploy the application to AWS Elastic Beanstalk. Expose the AWS Lambda functionality by using an Application Load Balancer.

Answer: (SHOW ANSWER)

Requirement Summary:

Simple REST endpoint for weather data

Light backend usage (POC, testing)

Wants caching support to reduce backend calls

Must be cost-effective

Evaluate Options:

B: Lambda + API Gateway + SAM

Serverless = No idle costs

API Gateway can enable caching (response caching at endpoint level)

SAM makes deployment simple and repeatable

Perfect for low-traffic testing

A: EKS + API Gateway

High overhead

Not cost-effective for POC/testing

C: ECS + API Gateway

Similar to A: Container orchestration not needed for light REST endpoint D: Elastic Beanstalk + ALB + Lambda Overly complex and does not directly expose Lambda Beanstalk better suited for full apps, not small REST functions AWS SAM: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.

html

API Gateway caching: https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html

Serverless Best Practices: https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html

NEW QUESTION: 16

A developer is writing an application that processes data delivered into an Amazon S3 bucket. The data is delivered approximately 10 times per day, and the developer expects the processing to complete in less than
1 minute on average.
How can the developer deploy and invoke the application with the LOWEST cost and LOWEST latency?

A. Deploy the application as an AWS Lambda function and invoke it by using an Amazon CloudWatch alarm that is triggered by an S3 object upload.

B. Deploy the application as an AWS Lambda function and invoke it by using an Amazon S3 event notification.

C. Deploy the application as an AWS Lambda function and invoke it by using an Amazon CloudWatch scheduled event.

D. Deploy the application on an Amazon EC2 instance and poll the S3 bucket for new objects.

Answer: B (LEAVE A REPLY)

AWS Lambda combined with Amazon S3 event notifications provides the most cost-effective and lowest- latency solution for processing objects uploaded to S3. AWS documentation states that S3 event notifications can invoke Lambda functions immediately after object creation, eliminating delays associated with polling or scheduled execution. Because the data arrives only about 10 times per day and processing completes quickly, Lambda is ideal due to its pay-per-invocation pricing model. There is no need to maintain always-on infrastructure, which significantly reduces cost compared to running EC2 instances. The developer pays only for execution time and requests.

Using CloudWatch alarms (Option A) or scheduled events (Option C) introduces unnecessary latency and complexity. These approaches either delay processing or require periodic checks rather than reacting instantly to object uploads. Polling from EC2 (Option D) is the most expensive and least efficient option because it requires continuously running compute resources.

AWS best practices for event-driven architectures explicitly recommend S3 event notifications # Lambda for near-real-time processing of uploaded data. This approach minimizes operational overhead, achieves near- zero idle cost, and provides the fastest possible response to new data.

Therefore, invoking a Lambda function directly through an S3 event notification is the optimal solution.

NEW QUESTION: 17
A company has an Amazon API Gateway REST API that integrates with an AWS Lambda function. The API' 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 (LEAVE A REPLY)
Amazon API Gateway stages are designed to represent different deployment environments such as development, testing, and production. AWS documentation recommends using separate stages combined with stage variables to reference different backend resources without duplicating API definitions.
In this scenario, the development stage already uses a stage variable to reference the Lambda dev alias. To expose the production alias safely, the developer should deploy the API to a new stage named production and configure a stage variable that points to the Lambda prod alias. This approach allows the same API configuration and methods to be reused while cleanly separating environments.
Stage variables are resolved at runtime and are commonly used to reference Lambda aliases in integration ARNs. This enables controlled promotion of code from development to production without modifying methods or integrations.
Creating new methods (Options A and B) is unnecessary and increases operational complexity. Directly hardcoding the Lambda alias in the integration (Option D) removes flexibility and deviates from AWS best practices for environment separation.
Therefore, deploying a new API stage and using stage variables to reference the prod Lambda alias is the correct and most efficient solution.

NEW QUESTION: 18
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 (LEAVE A REPLY)
This solution meets the requirements because it provides a caching layer that can store and retrieve encrypted data from multiple nodes. Amazon ElastiCache for Redis supports encryption at rest and in transit, and can scale horizontally to increase the cache capacity and availability. Amazon ElastiCache for Memcached does not support encryption, Amazon CloudFront is a content delivery network that is not suitable for caching database queries, and Amazon DynamoDB Accelerator (DAX) is a caching service that only works with DynamoDB tables.
Reference: [Amazon ElastiCache for Redis Features], [Choosing a Cluster Engine]

NEW QUESTION: 19
A company wants to ensure that only one user from its Admin group has the permanent right to delete an Amazon EC2 resource. The company must not modify the existing Admin group policy.
What should a developer use to meet these requirements?
A. AWS managed policy
B. Inline policy
C. IAM trust relationship
D. AWS STS
Answer: B (LEAVE A REPLY)
An inline IAM policy is directly attached to a specific IAM user, group, or role and applies only to that principal. AWS documentation states that inline policies are useful when permissions should be tightly scoped and not reused.
In this scenario, the Admin group policy cannot be changed, but one specific user needs permanent delete permissions. Attaching an inline policy directly to that user grants the required permissions without impacting other Admin users.
AWS managed policies (Option A) are reusable and not suitable for user-specific access. IAM trust relationships (Option C) control role assumption, not resource permissions. AWS STS (Option D) provides temporary credentials, not permanent access.
Therefore, an inline policy is the correct solution.

NEW QUESTION: 20
A developer manages an application that stores user objects in an Amazon S3 bucket without versioning enabled. The application has premium users and basic users.

After premium users upload objects, the premium users have unlimited downloads of their objects. Their objects are stored with a premium/ prefix. After basic users upload objects, the basic users can download their objects for 90 days. Their objects are stored with a basic/ prefix.
The developer needs to implement a solution to automatically delete objects for the basic users after 90 days.
Which solution will meet these requirements with the LEAST development effort?
A. Create an AWS Lambda function that removes any objects in the S3 bucket that have the basic/ prefix and are more than 90 days old. Set up an Amazon EventBridge schedule to invoke the Lambda function every day.
B. Set up an S3 Lifecycle rule that applies to the objects that have the premium/ prefix. Set the S3 Lifecycle rule action to expire the current version of the objects that have the premium/ prefix after 90 days.
C. Set up an S3 Lifecycle rule that applies to the objects that have the basic/ prefix. Set the S3 Lifecycle rule action to expire the current version of the objects that have the basic/ prefix after 90 days.
D. Create a rule for the S3 bucket to identify objects that have the basic/ prefix. Set the rule action to delete any objects that have object delete markers and unfinished multipart uploads after 90 days.
Answer: C (LEAVE A REPLY)
The requirement is to automatically delete objects for basic users after 90 days with the least development effort. Amazon S3 provides a built-in, fully managed feature for this: S3 Lifecycle configuration rules.
Lifecycle rules can filter objects by prefix (and/or tags) and then perform actions such as expiring (deleting) objects after a specified number of days since creation. Because the bucket does not have versioning enabled, the relevant action is to expire the current object after the retention period.
Option C matches this exactly: create an S3 Lifecycle rule that applies only to objects under the basic/ prefix and configure the rule to expire current versions after 90 days. S3 then automatically and continuously enforces the policy without any code, scheduling, or operational maintenance. This is typically the lowest- effort and most reliable approach because it is handled by S3 itself.
Option A requires writing and operating a Lambda function plus a scheduler, handling pagination, permissions, error handling, retries, and potential costs and throttling when scanning large buckets. That is more development and operational effort than a lifecycle rule. Option B is incorrect because it targets the premium/ prefix, but premium objects must be retained indefinitely. Option D refers to delete markers and unfinished multipart uploads; delete markers are relevant primarily for versioned buckets, and cleaning up multipart uploads does not implement the required "delete basic objects after 90 days" behavior.

Therefore, the correct solution is C: use an S3 Lifecycle rule scoped to the basic/ prefix to expire objects after
90 days, achieving automated retention with minimal development effort.

NEW QUESTION: 21
A company uses a custom root certificate authority certificate chain (Root CA Cert) that is 10 KB in size generate SSL certificates for its on-premises HTTPS endpoints. One of the company's cloud based applications has hundreds of AWS Lambda functions that pull date from these endpoints. A developer updated the trust store of the Lambda execution environment to use the Root CA Cert when the Lambda execution environment is initialized. The developer bundled the Root CA Cert as a text file in the Lambdas deployment bundle.
After 3 months of development the root CA Cert is no longer valid and must be updated. The developer needs a more efficient solution to update the Root CA Cert for all deployed Lambda functions. The solution must not include rebuilding or updating all Lambda functions that use the Root CA Cert. The solution must also work for all development, testing and production environment. Each environment is managed in a separate AWS account.
When combination of steps Would the developer take to meet these environments MOST cost-effectively?
(Select TWO)
A. Store the Root CA Cert as a secret in AWS Secrets Manager. Create a resource-based policy. Add IAM users to allow access to the secret
B. Store the Root CA Cert as a Secure Sting parameter in aws Systems Manager Parameter Store Create a resource-based policy. Add IAM users to allow access to the policy.
C. Store the Root CA Cert in an Amazon S3 bucket. Create a resource- based policy to allow access to the bucket.
D. Refactor the Lambda code to load the Root CA Cert from the Root CA Certs location. Modify the runtime trust store inside the Lambda function handler.
E. Refactor the Lambda code to load the Root CA Cert from the Root CA Cert's location. Modify the runtime trust store outside the Lambda function handler.
Answer: (SHOW ANSWER)
This solution will meet the requirements by storing the Root CA Cert as a Secure String parameter in AWS Systems Manager Parameter Store, which is a secure and scalable service for storing and managing configuration data and secrets. The resource-based policy will allow IAM users in different AWS accounts and environments to access the parameter without requiring cross-account roles or permissions. The Lambda code will be refactored to load the Root CA Cert from the parameter store and modify the runtime trust store outside the Lambda function handler, which will improve performance and reduce

latency by avoiding repeated calls to Parameter Store and trust store modifications for each invocation of the Lambda function.
Option A is not optimal because it will use AWS Secrets Manager instead of AWS Systems Manager Parameter Store, which will incur additional costs and complexity for storing and managing a non-secret configuration data such as Root CA Cert. Option C is not optimal because it will deactivate the application secrets and monitor the application error logs temporarily, which will cause application downtime and potential data loss. Option D is not optimal because it will modify the runtime trust store inside the Lambda function handler, which will degrade performance and increase latency by repeating unnecessary operations for each invocation of the Lambda function.
References: AWS Systems Manager Parameter Store, [Using SSL/TLS to Encrypt a Connection to a DB Instance]

NEW QUESTION: 22
A developer is building an application on a fleet of Amazon EC2 Linux instances that run the Apache web server. The application must send API calls that contain sensitive customer data to a second fleet of Linux instances that also run Apache. The two fleets are deployed in peered VPCs within the same AWS account and AWS Region.
All sensitive data must be encrypted in transit.
Which solution will meet these requirements in the MOST operationally efficient way?
A. Create security groups in each VPC that allow traffic only from the other fleet's security group.
B. Create an AWS Site-to-Site VPN connection between the two peered VPCs and route the API traffic through the VPN.
C. Encrypt all Amazon EBS volumes with a customer managed AWS KMS key and attach an IAM instance profile that allows access to the key.
D. Request a certificate through AWS Certificate Manager (ACM) and redeploy both fleets by using TLS for Apache with the ACM-issued certificate.
Answer: D (LEAVE A REPLY)
The requirement in this scenario is encryption in transit for sensitive data exchanged between two EC2- based application fleets. AWS best practices clearly distinguish between network isolation and transport- layer encryption. Security groups (Option A) restrict traffic but do not encrypt it. EBS encryption (Option C) protects data at rest and does not affect data transmitted over the network.
Although a Site-to-Site VPN (Option B) would encrypt traffic, AWS documentation considers this approach unnecessary and operationally heavy when both workloads run inside AWS and application-level encryption is sufficient.
The most efficient and AWS-recommended approach is to use TLS (HTTPS) for application communication.
AWS Certificate Manager (ACM) allows developers to provision and manage TLS certificates without manual certificate handling. Apache can be configured to use HTTPS

with ACM-issued certificates, ensuring that all API traffic between the fleets is encrypted in transit using industry-standard TLS.
AWS documentation consistently recommends TLS for service-to-service communication within AWS when sensitive data is transmitted. This approach minimizes operational overhead, avoids additional networking infrastructure, and integrates natively with existing EC2-based applications.
Therefore, using ACM-issued certificates and HTTPS for Apache communication is the correct and most efficient solution.

NEW QUESTION: 23
A developer is creating an ecommerce workflow in an AWS Step Functions state machine that includes a HTTP Task state. The task passes shipping information and order details to an endpoint.
The developer needs to test the workflow to confirm that the HTTP headers and body are correct and that the responses meet expectations.
A. Use the TestState API to invoke only the HTTP Task. Set the inspection level to TRACE.
B. Use the TestState API to invoke the state machine. Set the inspection level to DEBUG.
C. Use the data flow simulator to invoke only the HTTP Task. View the request and response data.
D. Change the log level of the state machine to ALL. Run the state machine.
Answer: A (LEAVE A REPLY)
Comprehensive and Detailed Step-by-Step Explanation:
To confirm that the HTTP headers, body, and responses meet expectations, you need to test the specific HTTP Task state in isolation and inspect the details.
Option A: TestState API with TRACE:
The TestState API allows developers to test individual states in a state machine without executing the entire workflow.
Setting the inspection level to TRACE provides detailed information about the HTTP request and response, including headers, body, and status codes.
This option provides the precise and granular testing required to verify the HTTP Task functionality.
Why Other Options Are Incorrect:
Option B: The DEBUG inspection level provides less detailed information than TRACE and focuses on general debugging, not a detailed view of HTTP interactions.
Option C: Step Functions does not have a "data flow simulator" to test individual tasks; this option is not valid.
Option D: Changing the state machine's log level to ALL increases logging granularity for the entire state machine but does not allow isolated testing of a specific HTTP Task.
References:
AWS Step Functions: Testing State Machines

NEW QUESTION: 24
A company has an existing application that has hardcoded database credentials A developer needs to modify the existing application The application is deployed in two AWS Regions with an active-passive failover configuration to meet company's disaster recovery strategy The developer needs a solution to store the credentials outside the code. The solution must comply With the company's disaster recovery strategy Which solution Will meet these requirements in the MOST secure way?
A. Store the credentials in AWS Secrets Manager in the primary Region. Enable secret replication to the secondary Region Update the application to use the Amazon Resource Name (ARN) based on the Region.
B. Store credentials in AWS Systems Manager Parameter Store in the primary Region. Enable parameter replication to the secondary Region. Update the application to use the Amazon Resource Name (ARN) based on the Region.
C. Store credentials in a config file. Upload the config file to an S3 bucket in me primary Region. Enable Cross-Region Replication (CRR) to an S3 bucket in the secondary region. Update the application to access the config file from the S3 bucket based on the Region.
D. Store credentials in a config file. Upload the config file to an Amazon Elastic File System (Amazon EFS) file system. Update the application to use the Amazon EFS file system Regional endpoints to access the config file in the primary and secondary Regions.
Answer: (SHOW ANSWER)
AWS Secrets Manager is a service that allows you to store and manage secrets, such as database credentials, API keys, and passwords, in a secure and centralized way. It also provides features such as automatic secret rotation, auditing, and monitoring1. By using AWS Secrets Manager, you can avoid hardcoding credentials in your code, which is a bad security practice and makes it difficult to update them. You can also replicate your secrets to another Region, which is useful for disaster recovery purposes2. To access your secrets from your application, you can use the ARN of the secret, which is a unique identifier that includes the Region name. This way, your application can use the appropriate secret based on the Region where it is deployed3.
AWS Secrets Manager
Replicating and sharing secrets
Using your own encryption keys

NEW QUESTION: 25
A company requires that all applications running on Amazon EC2 use 1AM roles to gam access to AWS services. A developer is modifying an application that currently relies on 1AM user access keys stored in environment variables to access Amazon DynamoDB tables using boto, the AWS SDK for Python.
The developer associated a role with the same permissions as the 1AM user to the EC2 instance, then deleted the 1AM user. When the application was restarted, the AWS Access

Denied Exception messages started appearing in the application logs. The developer was able to use their personal account on the server to run DynamoDB API commands using the AWS CLI.
What is the MOST likely cause of the exception?
A. The AWS SDK does not support credentials obtained using an instance role.
B. The instance's security group does not allow access to http://169.254.169.254.
C. 1AM policies might take a few minutes to propagate to resources.
D. Disabled environment variable credentials are still being used by the application.
Answer: D (LEAVE A REPLY)

NEW QUESTION: 26
A developer uses AWS IAM Identity Center to interact with the AWS CLI and AWS SDKs on a local workstation. API calls to AWS services were working when the SSO access was first configured. However, the developer is now receiving Access Denied errors. The developer has not changed any configuration files or scripts that were previously working on the workstation.
What is the MOST likely cause of the developer's access issue?
A. The access permissions to the developer's AWS CLI binary file have changed.
B. The permission set that is assumed by IAM Identity Center does not have the necessary permissions to complete the API call.
C. The credentials from the IAM Identity Center federated role have expired.
D. The developer is attempting to make API calls to the incorrect AWS account.
Answer: C (LEAVE A REPLY)
Requirement Summary:
Developer uses AWS IAM Identity Center (SSO) with AWS CLI / SDKs
Initially working fine
Now receiving AccessDenied errors
No changes to config or scripts
Key Understanding:
IAM Identity Center credentials are temporary and time-limited. When you use SSO-based access via the AWS CLI (aws sso login), it obtains temporary credentials stored in the local cache.
By default, these expire in 1 hour (can be extended).
Evaluate Options:
A). Permissions to CLI binary changed
Unlikely - this would produce execution errors, not AccessDenied from AWS API B). Permission set lacks required permissions Then the error would have occurred from the beginning, not after time C). IAM Identity Center credentials expired Most likely - user hasn't refreshed credentials using aws sso login again After expiration, API calls fail with AccessDenied D). Developer is calling the wrong AWS account Would likely show different types of errors (AccountNotFound, etc.) SSO and AWS CLI:

https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-sso.html Troubleshooting SSO CLI: https://docs.aws.amazon.com/cli/latest/userguide/sso-configure-profile-token.html

NEW QUESTION: 27
A company created an application to consume and process data. The application uses Amazon SQS and AWS Lambda functions. The application is currently working as expected, but it occasionally receives several messages that it cannot process properly. The company needs to clear these messages to prevent the queue from becoming blocked. A developer must implement a solution that makes queue processing always operational. The solution must give the company the ability to defer the messages with errors and save these messages for further analysis. What is the MOST operationally efficient solution that meets these requirements?
A. Configure Amazon CloudWatch Logs to save the error messages to a separate log stream.
B. Create a new SQS queue. Set the new queue as a dead-letter queue for the application queue. Configure the Maximum Receives setting.
C. Change the SQS queue to a FIFO queue. Configure the message retention period to 0 seconds.
D. Configure an Amazon CloudWatch alarm for Lambda function errors. Publish messages to an Amazon SNS topic to notify administrator users.
Answer: (SHOW ANSWER)
Using a dead-letter queue (DLQ) with Amazon SQS is the most operationally efficient solution for handling unprocessable messages.
Amazon SQS Dead-Letter Queue:
A DLQ is used to capture messages that fail processing after a specified number of attempts.
Allows the application to continue processing other messages without being blocked.
Messages in the DLQ can be analyzed later for debugging and resolution.
Why DLQ is the Best Option:
Operational Efficiency: Automatically defers messages with errors, ensuring the queue is not blocked.
Analysis Ready: Messages in the DLQ can be inspected to identify recurring issues.
Scalable: Works seamlessly with Lambda and SQS at scale.
Why Not Other Options:
Option A: Logs the messages but does not resolve the queue blockage issue.
Option C: FIFO queues and 0-second retention do not provide error handling or analysis capabilities.
Option D: Alerts administrators but does not handle or store the unprocessable messages.
Steps to Implement:
Create a new SQS queue to serve as the DLQ.

Attach the DLQ to the primary queue and configure the Maximum Receives setting.
Using Amazon SQS Dead-Letter Queues
Best Practices for Using Amazon SQS with AWS Lambda

NEW QUESTION: 28
A software company is launching a multimedia application. The application will allow guest users to access sample content before the users decide if they want to create an account to gain full access. The company wants to implement an authentication process that can identify users who have already created an account.
The company also needs to keep track of the number of guest users who eventually create an account.
Which combination of steps will meet these requirements? {Select TWO.)
A. Create a role for authenticated users that allows access to all content. Create a role for unauthenticated users that allows access to only the sample content.
B. Allow all users to access the sample content by default. Create a role for authenticated users that allows access to the other content.
C. Create an Amazon Cognito user pool. Configure the user pool to allow unauthenticated users. Exchange user tokens for temporary credentials that allow authenticated users to assume a role.
D. Create an Amazon CloudFront distribution. Configure the distribution to allow unauthenticated users.
Exchange user tokens for temporary credentials that allow all users to assume a role.
E. Create an Amazon Cognito identity pool. Configure the identity pool to allow unauthenticated users.
Exchange unique identity for temporary credentials that allow all users to assume a role.
Answer: (SHOW ANSWER)

NEW QUESTION: 29
An ecommerce company is planning to migrate an on-premises Microsoft SQL Server database to the AWS Cloud. The company needs to migrate the database to SQL Server Always On availability groups. The cloud- based solution must be highly available.
Which solution will meet these requirements?
A. Deploy three Amazon EC2 instances with SQL Server across three Availability Zones. Attach one Amazon Elastic Block Store (Amazon EBS) volume to the EC2 instances.
B. Migrate the database to Amazon RDS for SQL Server. Configure a Multi-AZ deployment and read replicas.
C. Deploy three Amazon EC2 instances with SQL Server across three Availability Zones. Use Amazon FSx for Windows File Server as the storage tier.
D. Deploy three Amazon EC2 instances with SQL Server across three Availability Zones. Use Amazon S3 as the storage tier.
Answer: C (LEAVE A REPLY)

Why Option C is Correct:SQL Server Always On availability groups require a shared storage solution.
Amazon FSx for Windows File Server provides the shared storage necessary to implement Always On availability groups in a highly available configuration.
Why Other Options are Incorrect:
Option A: A single EBS volume cannot provide shared storage for Always On availability groups.
Option B: RDS does not support SQL Server Always On availability groups.
Option D: S3 is not a suitable storage tier for SQL Server database operations.
AWS Documentation References:
Amazon FSx for Windows File Server

NEW QUESTION: 30
A developer is modifying an AWS Lambda function that accesses an Amazon RDS for MySQL database. The developer discovers that the Lambda function has the database credentials stored as plaintext in the Lambda function code.
The developer must implement a solution to make the credentials more secure. The solution must include automated credential rotation every 30 days.
Which solution will meet these requirements?
A. Move the credentials to a secret in AWS Secrets Manager. Modify the Lambda function to read from Secrets Manager. Set a schedule to rotate the secret every 30 days.
B. Move the credentials to a secure string parameter in AWS Systems Manager Parameter Store. Modify the Lambda function to read from Parameter Store. Set a schedule to rotate the parameter every 30 days.
C. Move the credentials to an encrypted Amazon S3 bucket. Modify the Lambda function to read from the S3 bucket. Configure S3 Object Lambda to rotate the credentials every 30 days.
D. Move the credentials to a secure string parameter in AWS Systems Manager Parameter Store. Create an Amazon EventBridge rule to rotate the parameter every 30 days.
Answer: A (LEAVE A REPLY)
Requirement Summary:
Lambda function accesses RDS for MySQL
Credentials are currently hardcoded in code (insecure)
Must enable automated credential rotation every 30 days
Option A: Use AWS Secrets Manager + automatic rotation
Best and secure option
Secrets Manager allows:
Secure storage of secrets
Integration with RDS for automatic rotation
Scheduled rotation every X days (e.g., 30 days)
Lambda can fetch credentials via SDK (GetSecretValue)

Option B: Use SSM Parameter Store + rotation
SSM does not support automatic rotation of secrets.
You'd need to build custom rotation logic = higher operational overhead.
Option C: Encrypted S3 + Object Lambda rotation
Not intended for credential storage.
S3 is not a secrets management system, and Object Lambda does not perform rotation.
Option D: SSM + EventBridge rotation
No native integration between Parameter Store and EventBridge for secret rotation.
You'd need to build custom Lambda functions = higher maintenance.
Secrets Manager rotation for RDS:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating- secrets.html
Secure retrieval in Lambda:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets_lambda.html Integration with RDS MySQL:
https://docs.aws.amazon.com/secretsmanager/latest/userguide
/integrating_rds_config.html

NEW QUESTION: 31
A developer is building an application on AWS. The application has an Amazon API Gateway API that sends requests to an AWS Lambda function. The API is experiencing increased latency because the Lambda function has limited available CPU to fulfill the requests.
Before the developer deploys the API into production, the developer must configure the Lambda function to have more CPU.
Which solution will meet this requirement?
A. Increase the virtual CPU (vCPU) cores quota of the Lambda function.
B. Increase the amount of memory that is allocated to the Lambda function.
C. Increase the ephemeral storage size of the Lambda function.
D. Increase the timeout value of the Lambda function.
Answer: B (LEAVE A REPLY)
In AWS Lambda, CPU power scales proportionally with the memory configuration. Lambda does not let you directly set "CPU cores" as a standalone setting in the general case; instead, increasing the function's configured memory increases the CPU allocation (and other resources such as network throughput) available to the function. Therefore, to reduce latency caused by insufficient CPU, the developer should increase the function's memory setting.
Option B directly addresses the CPU limitation in the supported Lambda configuration model. This is a common performance tuning approach: raise memory, benchmark, and find the optimal cost/performance point.

Option A is not the right lever for most Lambda functions because Lambda compute is configured via memory (and architecture), not by directly raising a "vCPU quota" per function in a typical tuning workflow.
Option C increases /tmp storage capacity and helps when dealing with large temporary files, not CPU availability.
Option D increases the maximum runtime allowed per invocation, but it does not give the function more CPU and will not reduce latency caused by CPU starvation.
Therefore, increasing the allocated memory is the correct way to increase CPU for a Lambda function.

NEW QUESTION: 32
A developer runs an application that displays scores for sports games on Amazon EC2 instances. The application uses a Redis client to retrieve the scores from an Amazon ElastiCache (Redis OSS) cluster.
The developer observes increased latency during operations on the cache because of connection failures to the cluster. The developer needs to resolve the latency issues.
A. Configure the Redis client to use an exponential backoff retry strategy to establish cache connections.
B. Store the scores in the application's memory. Perform bulk set operations on the scores that are stored in memory.
C. Configure the Redis client in the application to persist connections to the cluster by implementing a connection pool.
D. Deploy more nodes in the ElastiCache cluster. Update the Redis client to discover the new nodes.
Answer: C (LEAVE A REPLY)
* Why Option C is Correct:Implementing a connection pool in the Redis client reduces connection overhead and avoids frequent connection establishment, which helps to reduce latency and connection failures.
* Why Other Options are Incorrect:
* Option A: Exponential backoff retry strategies help with transient failures but do not resolve latency caused by frequent connection establishment.
* Option B: Storing scores in application memory adds complexity and risks inconsistency.

* Option D: Adding more nodes is unnecessary unless the cluster is under heavy load. Latency due to connection failures is better addressed at the application level.
* AWS Documentation References:
* Amazon ElastiCache Best Practices

NEW QUESTION: 33
A developer is modifying a large-scale IoT application that stores device telemetry data in an Amazon DynamoDB table. The telemetry data is valuable only for a limited time, but the application stores the data indefinitely. Data storage is slowing the application down. The developer needs a solution to improve the performance of the application.
Which solution will meet this requirement in the MOST operationally efficient way?
A. Create an AWS Lambda function to run an Amazon EventBridge job on a schedule to scan the DynamoDB table for old items and to delete them.
B. Archive old data in an Amazon S3 bucket. Set up an S3 Lifecycle policy to transition old data to a more cost-effective storage class.
C. Set a TTL attribute for the telemetry data. Activate TTL on the DynamoDB table.
D. Change the table to on-demand capacity mode.
Answer: C (LEAVE A REPLY)
The most operationally efficient way to remove time-bound data from a DynamoDB table is to use Time to Live (TTL). TTL is a DynamoDB feature that lets you define an attribute (typically a Unix epoch timestamp) that represents an item's expiration time. After the timestamp has passed, DynamoDB automatically marks the item as expired and later deletes it in the background. This directly satisfies the requirement because the telemetry is only valuable for a limited time, yet it is currently stored indefinitely.
Enabling TTL (option C) improves performance and operational efficiency by preventing the table from growing without bounds. As the item count grows, access patterns that rely on scans, large partitions, or indexes can degrade, and storage-related overhead increases. TTL helps keep the dataset "fresh" by automatically removing stale telemetry, reducing the amount of data the application must work around and decreasing overall storage footprint.
Option A is operationally heavier: scanning and deleting items with a scheduled Lambda introduces ongoing maintenance, costs, and risk of throttling (table scans can be expensive and disruptive). It also requires custom logic, error handling, and retries. Option B addresses cost optimization for archived data but does not fix the DynamoDB table size or the performance degradation caused by keeping old data in the table; archiving is useful, but you still need an efficient deletion mechanism from DynamoDB. Option D (on-demand capacity mode) changes how capacity is managed, not how much data is stored; it does not remove stale items and therefore does not address the core problem.
Therefore, the best and most operationally efficient solution is C: add a TTL attribute to each telemetry item and enable TTL on the DynamoDB table so DynamoDB automatically expires and deletes old telemetry.

NEW QUESTION: 34
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 (LEAVE A REPLY)
* Option B: The target group port in the ALB must match the port specified in the ECS task definition. If there is a mismatch, the ALB health check will fail since it cannot correctly route traffic to the container.
* Option E: The ALB listener must have a default action that forwards requests to the correct target group associated with the ECS service. If this configuration is missing, the health check will fail as no traffic is routed to the service.
* Option A is irrelevant to resolving health check issues since capacity providers relate to provisioning compute capacity.
* Option C (hosted zone) is not directly related to ALB health checks.
* Option D (redirecting traffic) is not related to ECS health check configurations.
Reference:AWS ECS Health Check Documentation

NEW QUESTION: 35
A company's application uses an Amazon API Gateway REST API and AWS Lambda functions to upload media files to and fetch media files from a standard Amazon S3 Standard bucket. The company runs a nightly job on an Amazon EC2 instance to create dashboards and other visualizations for application users. The job usually runs for 1 to 2 hours.
A developer observes request throttling while the function is running. The application generates multiple 429 exceptions in the Lambda function logs when files do not process successfully. The developer needs to resolve the issue and ensure that all of the application ingests all files.

Which solution will meet these requirements?

A. Enable S3 Transfer Acceleration on the bucket. Use the appropriate endpoint.

B. Call the CreateMultipartUpload API in the Lambda functions to upload the files in pieces.

C. Implement the retry with a backoff pattern in the Lambda functions.

D. Set up an S3 Lifecycle policy to automatically move the media files to the S3 Intelligent-Tiering storage class.

Answer: C (LEAVE A REPLY)

HTTP 429 errors indicate throttling ("Too Many Requests"). In this architecture, throttling can occur at multiple layers (API Gateway rate limits, downstream AWS service API throttles, or dependency throttling).

Regardless of which service is throttling, the correct resilience pattern is to implement retries with exponential backoff and jitter for retryable failures. AWS SDKs and AWS best practices recommend backoff to reduce contention and allow the system to recover, while ensuring requests eventually succeed.

Option C directly addresses the problem and the requirement "ensure that all of the application ingests all files." With a retry + backoff pattern, transient throttling is handled gracefully: failed operations are retried after increasing delays, avoiding immediate retry storms that worsen throttling. This increases successful completion rate without requiring major architectural changes.

Option A (Transfer Acceleration) can improve upload latency from geographically distributed clients, but it does not resolve request throttling (429) caused by API limits.

Option B (multipart upload) helps with large object uploads and can improve throughput/reliability for big files, but it does not inherently prevent 429 throttling at API Gateway or other throttled calls.

Option D (Intelligent-Tiering) affects storage cost optimization, not ingestion throttling.

NEW QUESTION: 36

A real-time messaging application uses Amazon API Gateway WebSocket APIs with backend HTTP service.

A developer needs to build a feature in the application to identify a client that keeps connecting to and disconnecting from the WebSocket connection. The developer also needs the ability to remove the client Which combination of changes should the developer make to the application to meet these requirements?

(Select TWO.)

A. Switch to HTTP APIs in the backend service.

B. Switch to REST APIs in the backend service.

C. Use the callback URL to disconnect the client from the backend service.

D. Add code to track the client status in Amazon ElastiCache in the backend service.

E. Implement $connect and $disconnect routes in the backend service.

Answer: D,E (LEAVE A REPLY)

Requirement Summary:
WebSocket-based messaging app using API Gateway WebSocket APIs
Need to:
Identify clients repeatedly connecting/disconnecting
Be able to remove problematic clients
Evaluate Options:
A). Switch to HTTP APIs
HTTP APIs don't support WebSocket connections
B). Switch to REST APIs
REST APIs are not compatible with WebSockets
C). Use the callback URL to disconnect clients
Possible, but not a direct option
Callback URLs are used for sending messages to connected clients, not for disconnecting
D). Track client status in ElastiCache Good solution: Store and update connection state (connected, disconnected, timestamps) Helps track abuse or reconnections E). Implement $connect and $disconnect routes Required to capture connection lifecycle events These can be used to log/store client behavior and decide on removal WebSocket routes in API Gateway: https://docs.aws.amazon.com/apigateway/latest/developerguide /apigateway-websocket-api-route-selection.html
Managing WebSocket connections:
https://docs.aws.amazon.com/apigateway/latest/developerguide /apigateway-websocket-api-mapping-template-reference.html

NEW QUESTION: 37
A developer has observed an increase in bugs in the AWS Lambda functions that a development team has deployed in its Node.js application.
To minimize these bugs, the developer wants to implement automated testing of Lambda functions in an environment that closely simulates the Lambda environment.
The developer needs to give other developers the ability to run the tests locally. The developer also needs to integrate the tests into the team's continuous integration and continuous delivery (CI/CD) pipeline before the AWS Cloud Development Kit (AWS CDK) deployment.
Which solution will meet these requirements?
A. Create sample events based on the Lambda documentation. Create automated test scripts that use the cdk local invoke command to invoke the Lambda functions. Check the response. Document the test scripts for the other developers on the team. Update the CI/CD pipeline to run the test scripts.
B. Install a unit testing framework that reproduces the Lambda execution environment. Create sample events based on the Lambda documentation. Invoke the handler function by using a unit testing framework. Check the response. Document how to run the unit

testing framework for the other developers on the team. Update the CI/CD pipeline to run the unit testing framework.
C. Install the AWS Serverless Application Model (AWS SAM) CLI tool. Use the sam local generate-event command to generate sample events for the automated tests. Create automated test scripts that use the sam local invoke command to invoke the Lambda functions. Check the response. Document the test scripts for the other developers on the team. Update the CI/CD pipeline to run the test scripts.
D. Create sample events based on the Lambda documentation. Create a Docker container from the Node.js base image to invoke the Lambda functions. Check the response. Document how to run the Docker container for the other developers on the team. Update the CIICD pipeline to run the Docker container.
Answer: C (LEAVE A REPLY)
The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) is a command-line tool for local development and testing of Serverless applications3. The sam local generate-event command of AWS SAM CLI generates sample events for automated tests3. The sam local invoke command is used to invoke Lambda functions3. Therefore, option C is correct.

NEW QUESTION: 38
A developer is testing an AWS Lambda function by using the AWS SAM local CLI. The application that is implemented by the Lambda function makes several AWS API calls by using the AWS SDK. The developer wants to allow the function to make AWS API calls in a test AWS account from the developer's laptop.
What should the developer do to meet these requirements?
A. Edit the template.yml file. Add the AWS_ACCESS_KEY_ID property and the AWS_SECRET_ACCESS_KEY property in the Globals section.
B. Add a test profile by using the aws configure command with the --profile option. Run AWS SAM by using sam local invoke with the --profile option.
C. Edit the template.yml file. For the AWS::Serverless::Function resource, set the role to an IAM role in the AWS account.
D. Run the function by using sam local invoke. Override the AWS_ACCESS_KEY_ID parameter and the AWS_SECRET_ACCESS_KEY parameter by specifying the --parameter-overrides option.
Answer: (SHOW ANSWER)
When using AWS SAM local (sam local invoke, sam local start-api), the Lambda code runs in a local container on the developer's machine. If that code uses the AWS SDK to call AWS services, it needs AWS credentials available in the local environment. The recommended way is to rely on the AWS SDK's standard credential provider chain, which includes credentials stored by the AWS CLI in ~/.aws/credentials and selected via a named profile.

By running aws configure --profile <profileName>, the developer creates a dedicated set of access keys (and optionally a session token/region) for the test account. Then, running SAM with sam local invoke --profile
<profileName> causes SAM to supply those credentials to the local container so the function can authenticate to AWS and make real API calls in the intended test account. This approach is clean, repeatable, and avoids hardcoding secrets into templates or source code.
Option A is insecure and not the intended use of SAM templates; templates are infrastructure definitions, not secret stores. Option C sets the Lambda execution role for deployed functions in AWS, but it does not automatically grant credentials to code running locally on a laptop. Option D misuses --parameter-overrides, which is meant for overriding template parameters at deployment/build time, not for securely injecting AWS access keys into local execution.
Therefore, the correct approach is B: configure an AWS CLI profile and run SAM local with the --profile option so the Lambda code can call AWS APIs in the test account using standard credential handling.

NEW QUESTION: 39
An application uses AWS X-Ray to generate a large amount of trace data on an hourly basis. A developer wants to use filter expressions to limit the returned results through user-specified custom attributes.
How should the developer use filter expressions to filter the results in X-Ray?
A. Add custom attributes as annotations in the segment document.
B. Add custom attributes as metadata in the segment document.
C. Add custom attributes as new segment fields in the segment document.
D. Create new sampling rules that are based on custom attributes.
Answer: A (LEAVE A REPLY)
In AWS X-Ray, filter expressions can filter trace data based on indexed fields that X-Ray can query efficiently. Custom data can be added to segments in two main ways: annotations and metadata. The critical difference is that annotations are indexed and can be used for filtering, while metadata is not indexed and is intended for additional diagnostic context that you do not typically query on.
Therefore, if the developer wants user-specified custom attributes to be usable in X-Ray filter expressions, the developer should record those attributes as annotations in the segment document. Once recorded as annotations, the developer can write filter expressions that match annotation keys/values and return only the relevant traces.
Option B is incorrect because metadata is not indexed and cannot be used in filter expressions for trace search the same way annotations can.
Option C is incorrect because segment documents have a defined schema; adding arbitrary "new segment fields" is not the intended method for searchable custom attributes.

Option D is unrelated: sampling rules control which requests get traced in the first place. They are not used to filter returned results by custom attributes after traces are recorded.

NEW QUESTION: 40
An Amazon Simple Queue Service (Amazon SQS) queue serves as an event source for an AWS Lambda function In the SQS queue, each item corresponds to a video file that the Lambda function must convert to a smaller resolution The Lambda function is timing out on longer video files, but the Lambda function's timeout is already configured to its maximum value What should a developer do to avoid the timeouts without additional code changes'?
A. Increase the memory configuration of the Lambda function
B. Increase the visibility timeout on the SQS queue
C. Increase the instance size of the host that runs the Lambda function.
D. Use multi-threading for the conversion.
Answer: B (LEAVE A REPLY)
* Visibility Timeout: When an SQS message is processed by a consumer (here, the Lambda function), it's temporarily hidden from other consumers. Visibility timeout controls this duration.
* How It Helps:
* Increase the visibility timeout beyond the maximum processing time your Lambda might typically take for long videos.
* This prevents the message from reappearing in the queue while Lambda is still working, avoiding premature timeouts.
References:
SQS Visibility Timeout:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html

NEW QUESTION: 41
A developer is setting up AWS CodePipeline for a new application. During each build, the developer must generate a test report.
Which solution will meet this requirement?
A. Create an AWS CodeBuild build project that runs tests. Configure the buildspec file with the test report information.
B. Create an AWS CodeDeploy deployment that runs tests. Configure the AppSpec file with the test report information.
C. Run the builds on an Amazon EC2 instance that has AWS Systems Manager Agent (SSM Agent) installed and activated.
D. Create a repository in AWS CodeArtifact. Select the test report template.
Answer: (SHOW ANSWER)
In CodePipeline, the service designed to run builds, execute unit/integration tests, and produce build artifacts (including test reports) is AWS CodeBuild. CodeBuild runs

commands specified in a buildspec.yml file and supports reporting outputs such as test results (for example, JUnit XML), code coverage, and other build metadata.
Option A is correct because the developer can configure the CodeBuild project to run the test suite during the build phase and configure the buildspec to publish test report files. This integrates naturally into CodePipeline as a build stage, and the reports are generated consistently on each pipeline execution.
Option B is incorrect: CodeDeploy is for deploying applications (in-place/blue-green) and lifecycle hooks, not for standard build/test report generation.
Option C increases operational overhead by managing an EC2 build server, which is unnecessary when CodeBuild provides managed build environments.
Option D is unrelated: CodeArtifact is a package repository, not a test reporting solution.
Therefore, use CodeBuild and configure test reports via the buildspec.

NEW QUESTION: 42
A developer has an application that asynchronously invokes an AWS Lambda function. The developer wants to store messages that resulted in failed invocations of the Lambda function so that the application can retry the call later.
What should the developer do to accomplish this goal with the LEAST operational overhead?
A. Set up CloudWatch Logs log groups to filter and store the messages in an S3 bucket. Import the messages in Lambda. Run the Lambda function again.
B. Configure Amazon EventBridge to send the messages to Amazon SNS to initiate the Lambda function again.
C. Implement a dead-letter queue for discarded messages. Set the dead-letter queue as an event source for the Lambda function.
D. Send Amazon EventBridge events to an Amazon SQS queue. Configure the Lambda function to pull messages from the SQS queue. Run the Lambda function again.
Answer: C (LEAVE A REPLY)
For asynchronous Lambda invocations, AWS provides built-in failure handling options that require minimal code and minimal operational work. When an async invocation fails (after Lambda's internal retry behavior), the event can be sent to a dead-letter queue (DLQ) so it is not lost. A DLQ is the standard mechanism for capturing events that could not be processed successfully, preserving the original payload for later inspection and reprocessing.
Using a DLQ (typically Amazon SQS or Amazon SNS) gives durable storage of failed events and decouples failure recovery from the main execution path. The developer can later re-drive the messages by building a simple replay process, such as moving messages from the DLQ back to the original event source or invoking the function again with the stored payloads. This aligns with the requirement: "store messages that resulted in failed invocations so the application can retry later." Among the options, C is the only one that uses Lambda's native async failure capture mechanism. Options A and B

introduce unnecessary complexity and do not reliably store the original failed invocation payloads as a first-class workflow (CloudWatch Logs is not a message queue, and EventBridge#SNS doesn't automatically capture only failed events from Lambda async invocations). Option D changes the architecture to an SQS pull model for the Lambda function; while valid, it is more than needed and adds design/operational considerations (polling, batching, visibility timeouts, DLQ configuration on the queue, etc.) compared with simply enabling DLQ support for async invokes.
Therefore, the least operational overhead solution is C: configure a dead-letter queue for the Lambda function's asynchronous invocations so failed events are stored for later retry.

NEW QUESTION: 43
A developer needs to write an AWS CloudFormation template on a local machine and deploy a CloudFormation stack to AWS.
What must the developer do to complete these tasks?
A. Install the AWS CLI. Configure the AWS CLI by using an I AM user name and password.
B. Install an AWS software development kit (SDK). Configure the SDK by using an X.509 certificate.
C. Install the AWS CLI. Configure the AWS CLI by using an SSH key.
D. Install the AWS CLI. Configure the AWS CLI by using an 1AM user access key and secret key.
Answer: D (LEAVE A REPLY)

NEW QUESTION: 44
A company has an application that runs on Amazon EC2 instances. The application needs to use dynamic feature flags that will be shared with other applications. The application must poll on an interval for new feature flag values. The values must be cached when they are retrieved.
Which solution will meet these requirements in the MOST operationally efficient way?
A. Store the feature flag values in AWS Secrets Manager. Configure an Amazon ElastiCache node to cache the values by using a lazy loading strategy in the application. Update the application to poll for the values on an interval from ElastiCache.
B. Store the feature flag values in an Amazon DynamoDB table. Configure DynamoDB Accelerator (DAX) to cache the values by using a lazy loading strategy in the application. Update the application to poll for the values on an interval from DynamoDB.
C. Store the feature flag values in AWS AppConfig. Configure AWS AppConfig Agent on the EC2 instances to poll for the values on an interval. Update the application to retrieve the values from the AppConfig Agent localhost endpoint.
D. Store the feature flag values in AWS Systems Manager Parameter Store. Configure the application to poll on an interval. Configure the application to use the AWS SDK to retrieve the values from Parameter Store and to store the values in memory.

Answer: C (LEAVE A REPLY)
Feature flags are a classic configuration-management use case: values change over time, multiple applications share them, and clients should retrieve updates efficiently with local caching. AWS AppConfig (part of AWS Systems Manager) is purpose-built to deploy and manage application configuration and feature flags. It provides controlled rollout, validation, and centralized configuration management. For operational efficiency, AWS offers the AWS AppConfig Agent for compute environments such as EC2.
With option C, the AppConfig Agent runs on each EC2 instance and polls AppConfig on a configured interval for updates. The agent maintains a local cache and exposes the current configuration through a localhost HTTP endpoint. The application then reads the feature flag values from the local endpoint, which is fast and reduces direct calls to AWS APIs.
This meets both requirements: periodic polling for new values and caching once retrieved, while minimizing application changes and centralizing operational control in AppConfig.
Option D (Parameter Store + in-memory cache) can work, but it pushes more responsibility into each application: polling logic, caching behavior, error handling, and consistency.
Option A misuses Secrets Manager (intended for secrets, not dynamic flags) and adds ElastiCache operational overhead. Option B similarly adds DAX and DynamoDB infrastructure and is not a standard feature-flag pattern; it also requires the app to implement polling and caching logic, reducing operational efficiency.
Therefore, C is the most operationally efficient solution: store flags in AWS AppConfig, run the AppConfig Agent on EC2 to handle polling and caching, and have the application read flags from the agent's localhost endpoint.

NEW QUESTION: 45
A developer is trying get data from an Amazon DynamoDB table called demoman-table.
The developer configured the AWS CLI to use a specific IAM use's credentials and ran the following command.
The command returned errors and no rows were returned.
What is the MOST likely cause of these issues?
A. The command is incorrect; it should be rewritten to use put-item with a string argument
B. The developer needs to log a ticket with AWS Support to enable access to the demoman-table
C. Amazon DynamoOB cannot be accessed from the AWS CLI and needs to called via the REST API
D. The IAM user needs an associated policy with read access to demoman-table
Answer: D (LEAVE A REPLY)
This solution will most likely solve the issues because it will grant the IAM user the necessary permission to access the DynamoDB table using the AWS CLI command. The error message indicates that the IAM user does not have sufficient access rights to perform the scan operation on the table. Option A is not optimal because it will change the command to use put-item instead of scan, which will not achieve the desiredresult of

getting data from the table. Option B is not optimal because it will involve contacting AWS Support, which may not be necessary or efficient for this issue. Option C is not optimal because it will state that DynamoDB cannot be accessed from the AWS CLI, which is incorrect as DynamoDB supports AWS CLI commands.
References: AWS CLI for DynamoDB, [IAM Policies for DynamoDB]

NEW QUESTION: 46
A developer created a Node.js-based AWS Lambda function by using a container image of an AWS OS-only base image. There is a new security patch for Node.js that must be patched to the new Lambda function.
Which solution will meet this requirement?
A. Set the runtime update mode of the Lambda function to Auto.
B. Patch the runtime version by redeploying the same version of the Lambda function.
C. Rebuild the Lambda container code with the latest version of the AWS OS base image. Publish a new version of the Lambda function.
D. Rebuild the Lambda container code with the latest Node.js patch version. Publish a new version of the Lambda function.
Answer: D (LEAVE A REPLY)
* Why Option D is Correct:When using a container-based AWS Lambda function, you are responsible for updating the base image for runtime patches. Rebuilding the container with the latest Node.js patch version ensures the function is updated with the required security patches. Publishing a new version of the Lambda function makes the updated image available for use.
* Why Other Options are Incorrect:
* Option A: The runtime update mode applies to functions using AWS-managed runtimes, not container-based runtimes.
* Option B: Redeploying the same version of the Lambda function does not apply the security patch.
* Option C: Rebuilding with the AWS OS base image does not guarantee that the latest Node.js patch version is included.
* AWS Documentation References:
* Lambda Function Container Images
* Node.js Updates in AWS Lambda

NEW QUESTION: 47
A developer is creating an AWS Lambda function. The Lambda function needs an external library to connect to a third-party solution The external library is a collection of files with a total size of 100 MB The developer needs to make the external library available to the Lambda execution environment and reduce the Lambda package space Which solution will meet these requirements with the LEAST operational overhead?
A. Create a Lambda layer to store the external library Configure the Lambda function to use the layer
B. Create an Amazon S3 bucket Upload the external library into the S3 bucket. Mount the S3 bucket folder in the Lambda function Import the library by using the proper folder in the mount point.
C. Load the external library to the Lambda function's /tmp directory during deployment of the Lambda package. Import the library from the /tmp directory.
D. Create an Amazon Elastic File System (Amazon EFS) volume. Upload the external library to the EFS volume Mount the EFS volume in the Lambda function. Import the library by using the proper folder in the mount point.
Answer: A (LEAVE A REPLY)
* Lambda Layers: These are designed to package dependencies that you can share across functions.
* How to Use:
* Create a layer, upload your 100MB library as a zip.
* Attach the layer to your function.
* In your function code, import the library from the standard layer path.
References:
* Lambda Layers: https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html

NEW QUESTION: 48
A developer is building an API that uses an Amazon CloudFront distribution to forward requests to an AWS Lambda function URL. The developer must ensure that the function URL can be accessed only through the CloudFront distribution and not directly.
Which solution will meet this requirement?
A. Create a resource-based policy for the CloudFront distribution. Configure the policy to allow access to the function URL.
B. Configure a resource-based policy for the Lambda function to allow only the CloudFront distribution to access the function URL. Configure the distribution to use an origin access control (OAC) for requests to the function URL.

C. Create an IAM role that has permissions to invoke the function URL. Configure a service role that has a CloudFront trust policy and permissions to make requests to the function URL.
D. Configure a resource-based policy for the Lambda function to allow only the CloudFront distribution's IP address range to access the function.
Answer: B (LEAVE A REPLY)
To ensure a Lambda function URL is reachable only through CloudFront, the solution must (1) make CloudFront the only authorized caller, and (2) prevent direct public access to the function URL endpoint. The AWS-native way to do this is to use a resource-based policy on the Lambda function that permits invocation only from the specific CloudFront distribution, combined with CloudFront's Origin Access Control (OAC) to sign origin requests.
A Lambda function URL can be configured for IAM-based authorization, and Lambda supports resource- based permissions (via lambda:AddPermission) that restrict who can invoke it. By scoping the permission so that only the CloudFront distribution (identified by a source ARN / distribution identifier) is allowed, direct requests that do not come through CloudFront will be rejected. This enforces the requirement that the function URL cannot be accessed directly.
CloudFront OAC is the modern mechanism to securely access origins by having CloudFront sign the requests it sends to the origin. When CloudFront signs requests and the origin (here, the Lambda function URL) is configured to accept only authorized requests per its resource policy, CloudFront becomes the only viable access path.
Option A is incorrect because CloudFront does not use a "resource-based policy" to grant access to an origin in this way; the enforcement must happen at the origin. Option C is not how CloudFront accesses origins; CloudFront does not assume a customer IAM role to fetch origin content. Option D is fragile and not recommended: CloudFront IP ranges can change and are not intended to be used as a static allowlist for origin protection; additionally, IP-based controls do not provide the same strong identity-based authorization that OAC + resource policy provides.
Therefore, B is the correct solution: restrict the Lambda function URL with a Lambda resource-based policy and configure CloudFront with OAC so only CloudFront-originated requests are accepted.

NEW QUESTION: 49
A developer is working on an ecommerce website The developer wants to review server logs without logging in to each of the application servers individually. The website runs on multiple Amazon EC2 instances, is written in Python, and needs to be highly available How can the developer update the application to meet these requirements with MINIMUM changes?
A. Rewrite the application to be cloud native and to run on AWS Lambda, where the logs can be reviewed in Amazon CloudWatch

B. Set up centralized logging by using Amazon OpenSearch Service, Logstash, and OpenSearch Dashboards
C. Scale down the application to one larger EC2 instance where only one instance is recording logs
D. Install the unified Amazon CloudWatch agent on the EC2 instances Configure the agent to push the application logs to CloudWatch
Answer: D (LEAVE A REPLY)
Centralized Logging Benefits: Centralized logging is essential for operational visibility in scalable systems, especially those using multiple EC2 instances like our e-commerce website. CloudWatch provides this capability, along with other monitoring features.
CloudWatch Agent: This is the best way to send custom application logs from EC2 instances to CloudWatch.
Here's the process:
Install the CloudWatch agent on each EC2 instance.
Configure the agent with a configuration file, specifying:
Which log files to collect.
The format in which to send logs to CloudWatch (e.g., JSON).
The specific CloudWatch Logs log group and log stream for these logs.
Viewing and Analyzing Logs: Once the agent is pushing logs, use the CloudWatch Logs console or API:
View and search the logs across all instances.
Set up alarms based on log events.
Use CloudWatch Logs Insights for sophisticated queries and analysis.
References:
Amazon CloudWatch Logs: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html
Unified CloudWatch Agent: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AgentReference.html
CloudWatch Logs Insights: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html

NEW QUESTION: 50
A company has developed a new serverless application using AWS Lambda functions that will be deployed using the AWS Serverless Application Model (AWS SAM) CLI.
Which step should the developer complete prior to deploying the application?
A. Compress the application to a zip file and upload it into AWS Lambda.
B. Test the new AWS Lambda function by first tracing it m AWS X-Ray.
C. Bundle the serverless application using a SAM package.
D. Create the application environment using the eb create my-env command.

Answer: C (LEAVE A REPLY)
This step should be completed prior to deploying the application because it prepares the application artifacts for deployment. The AWS Serverless Application Model (AWS SAM) is a framework that simplifies building and deploying serverless applications on AWS. The AWS SAM CLI is a command-line tool that helps you create, test, and deploy serverless applications using AWS SAM templates. The sam package command bundles the application artifacts, such as Lambda function code and API definitions, and uploads them to an Amazon S3 bucket. The command also returns a CloudFormation template that is ready to be deployed with the sam deploy command. Compressing the application to a zip file and uploading it to AWS Lambda will not work because it does not use AWS SAM templates or CloudFormation. Testing the new Lambda function by first tracing it in AWS X-Ray will not prepare the application for deployment, but only monitor its performance and errors. Creating the application environment using the eb create my-env command will not work because it is a command for AWS Elastic Beanstalk, not AWS SAM.

NEW QUESTION: 51
A developer is modifying an existing AWS Lambda function White checking the code the developer notices hardcoded parameter various for an Amazon RDS for SQL Server user name password database host and port. There also are hardcoded parameter values for an Amazon DynamoOB table. an Amazon S3 bucket, and an Amazon Simple Notification Service (Amazon SNS) topic.
The developer wants to securely store the parameter values outside the code m an encrypted format and wants to turn on rotation for the credentials. The developer also wants to be able to reuse the parameter values from other applications and to update the parameter values without modifying code.
Which solution will meet these requirements with the LEAST operational overhead?
A. Create an RDS database secret in AWS Secrets Manager. Set the user name password, database, host and port. Turn on secret rotation. Create encrypted Lambda environment variables for the DynamoDB table, S3 bucket and SNS topic.
B. Create an RDS database secret in AWS Secrets Manager. Set the user name password, database, host and port. Turn on secret rotation. Create Secure String parameters in AWS Systems Manager Parameter Store for the DynamoDB table, S3 bucket and SNS topic.
C. Create RDS database parameters in AWS Systems Manager Parameter. Store for the user name password, database, host and port. Create encrypted Lambda environment variables for me DynamoDB table, S3 bucket, and SNS topic. Create a Lambda function and set the logic for the credentials rotation task Schedule the credentials rotation task in Amazon EventBridge.
D. Create RDS database parameters in AWS Systems Manager Parameter. Store for the user name password database, host, and port. Store the DynamoDB table. S3 bucket, and

SNS topic in Amazon S3 Create a Lambda function and set the logic for the credentials rotation Invoke the Lambda function on a schedule.

Answer: B (LEAVE A REPLY)

This solution will meet the requirements by using AWS Secrets Manager and AWS Systems Manager Parameter Store to securely store the parameter values outside the code in an encrypted format. AWS Secrets Manager is a service that helps protect secrets such as database credentials by encrypting them with AWS Key Management Service (AWS KMS) and enabling automatic rotation of secrets. The developer can create an RDS database secret in AWS Secrets Manager and set the user name, password, database, host, and port for accessing the RDS database. The developer can also turn on secret rotation, which will change the database credentials periodically according to a specified schedule or event. AWS Systems Manager Parameter Store is a service that provides secure and scalable storage for configuration data and secrets. The developer can create Secure String parameters in AWS Systems Manager Parameter Store for the DynamoDB table, S3 bucket, and SNS topic, which will encrypt them with AWS KMS. The developer can also reuse the parameter values from other applications and update them without modifying code. Option A is not optimal because it will create encrypted Lambda environment variables for the DynamoDB table, S3 bucket, and SNS topic, which may not be reusable or updatable without modifying code. Option C is not optimal because it will create RDS database parameters in AWS Systems Manager Parameter Store, which does not support automatic rotation of secrets. Option D is not optimal because it will store the DynamoDB table, S3 bucket, and SNS topic in Amazon S3, which may introduce additional costs and complexity for accessing configuration data.

References: AWS Secrets Manager, [AWS Systems Manager Parameter Store]

NEW QUESTION: 52

A developer is creating a serverless application that uses an AWS Lambda function The developer will use AWS CloudFormation to deploy the application The application will write logs to Amazon CloudWatch Logs The developer has created a log group in a CloudFormation template for the application to use The developer needs to modify the CloudFormation template to make the name of the log group available to the application at runtime Which solution will meet this requirement?

A. Use the AWS:Include transform in CloudFormation to provide the log group's name to the application

B. Pass the log group's name to the application in the user data section of the CloudFormation template.

C. Use the CloudFormation template's Mappings section to specify the log group's name for the application.

D. Pass the log group's Amazon Resource Name (ARN) as an environment variable to the Lambda function

Answer: (SHOW ANSWER)

CloudFormation and Lambda Environment Variables:
CloudFormation is an excellent tool to manage infrastructure as code, including the log group resource.
Lambda functions can access environment variables at runtime, making them a suitable way to pass configuration information like the log group ARN.
CloudFormation Template Modification:
In your CloudFormation template, define the log group resource.
In the Lambda function resource, add an Environment section:
YAML
Environment:
Variables:
LOG_GROUP_ARN: !Ref LogGroupResourceName
Use code with caution.
content_copy
The !Ref intrinsic function retrieves the log group's ARN, which CloudFormation generates during stack creation.
Using the ARN in Your Lambda Function:
Within your Lambda code, access the LOG_GROUP_ARN environment variable.
Configure your logging library (e.g., Python's logging module) to send logs to the specified log group.
References:
AWS Lambda Environment Variables:
https://docs.aws.amazon.com/lambda/latest/dg/configuration-envvars.
html
CloudFormation !Ref Intrinsic Function:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide
/intrinsic-function-reference-ref.html

NEW QUESTION: 53
A developer is building an application that uses an AWS Lambda function to process data. The application requires minimum latency. The Lambda function must have predictable function start times. All setup activities for the execution environment must happen before invocation of the Lambda function.
Which solution will meet these requirements?
A. Optimize the static initialization code that runs when a new execution environment is prepared for the first time. Decrease and compress the size of the Lambda function package and the imported libraries and dependencies.
B. Publish a new version of the Lambda function. Configure provisioned concurrency for the Lambda function with the required minimum number of execution environments.

C. Increase the memory of the Lambda function to the maximum amount. Configure an Amazon EventBridge rule to schedule invocations of the Lambda function every minute to keep the execution environment active.
D. Increase the reserved concurrency of the Lambda function to the maximum value for unreserved account concurrency. Run any setup activities manually before the initial invocation of the Lambda function.
Answer: (SHOW ANSWER)

NEW QUESTION: 54
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: (SHOW ANSWER)
To reduce the time it takes for new Auto Scaling instances to become ready, the most effective optimization is to bake the time-consuming installation and configuration steps into a custom AMI. By creating a prebuilt AMI that already contains the application binaries, dependencies, and baseline configuration, new instances start from an image that is already near "ready to serve." This minimizes lengthy bootstrapping and significantly shortens the time required to pass ALB health checks and enter service. Therefore, creating a prebuilt AMI (B) is a primary performance optimization for faster scaling events and lower deployment variability.
After creating the AMI, the Auto Scaling group must actually use it. Because the infrastructure is managed with AWS CloudFormation, the developer should update the launch template resource (or launch configuration equivalent) in the CloudFormation template to reference the new AMI ID. This ensures that all newly launched instances use the optimized image consistently and that the change is tracked and repeatable as part of infrastructure as code. That is option C.
The other options are less optimal for the stated goal. Installing and configuring the app at launch time via Systems Manager Run Command (D) or CloudFormation helper scripts (E) still performs the expensive work during instance initialization, which prolongs launch time and can create variance (for example, dependency download time, repo availability, or

transient network slowness). Option A (Marketplace AMI) could help only if a suitable prebuilt image exists for the exact custom application-which is unlikely for a proprietary application-and it also reduces control over the build and hardening process compared to maintaining an internal golden AMI.
Therefore, the best combination is B (bake the app into a custom AMI) and C (update the CloudFormation- managed launch template to use that AMI).

NEW QUESTION: 55
A developer is building a web application that uses Amazon API Gateway to expose an AWS Lambda function to process requests from clients. During testing, the developer notices that the API Gateway times out even though the Lambda function finishes under the set time limit.
Which of the following API Gateway metrics in Amazon CloudWatch can help the developer troubleshoot the issue? (Choose two.)
A. CacheHitCount
B. IntegrationLatency
C. CacheMissCount
D. Latency
E. Count
Answer: B,D (LEAVE A REPLY)
Amazon API Gateway is a service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. Amazon CloudWatch is a service that monitors AWS resources and applications. API Gateway provides several CloudWatch metrics to help developers troubleshoot issues with their APIs. Two of the metrics that can help the developer troubleshoot the issue of API Gateway timing out are:
* IntegrationLatency: This metric measures the time between when API Gateway relays a request to the backend and when it receives a response from the backend. A high value for this metric indicates that the backend is taking too long to respond and may cause API Gateway to time out.
* Latency: This metric measures the time between when API Gateway receives a request from a client and when it returns a response to the client. A high value for this metric indicates that either the integration latency is high or API Gateway is taking too long to process the request or response.
References:
* [What Is Amazon API Gateway? - Amazon API Gateway]
* [Amazon API Gateway Metrics and Dimensions - Amazon CloudWatch]
* [Troubleshooting API Errors - Amazon API Gateway]

NEW QUESTION: 56

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. Update the settings for the build project to use an Amazon S3 bucket for large numbers of environment variables
B. Add the export LC-_ALL" on _ US, tuft" command to the pre _ build section to ensure POSIX Localization.
C. Use AWS Systems Manager Parameter Store to store large numbers ot environment variables
D. Use Amazon Cognate to store key-value pairs for large numbers of environment variables
Answer: C (LEAVE A REPLY)

NEW QUESTION: 57
A company is using an Amazon API Gateway REST API endpoint as a webhook to publish events from an on-premises source control management (SCM) system to Amazon EventBridge. The company has configured an EventBridge rule to listen for the events and to control application deployment in a central AWS account. The company needs to receive the same events across multiple receiver AWS accounts.
How can a developer meet these requirements without changing the configuration of the SCM system?
A. Deploy the API Gateway REST API to all the receiver AWS accounts. Create as many SCM webhooks as the number of AWS accounts.
B. Convert the API Gateway type from REST API to HTTP API.
C. Grant permission to the central AWS account for EventBridge to access the receiver AWS accounts.
Add an EventBridge event bus on the receiver AWS accounts as the targets to the existing EventBridge rule.
D. Deploy the API Gateway REST API to all the required AWS accounts. Use the same custom domain name for all the gateway endpoints so that a single SCM webhook can be used for all events from all accounts.
Answer: (SHOW ANSWER)

NEW QUESTION: 58
A company runs applications on Amazon EKS containers. The company sends application logs from the containers to an Amazon CloudWatch Logs log group. The company needs to process log data in real time based on a specific error in the application logs. Which combination of steps will meet these requirements?
(Select TWO.)
A. Create an Amazon SNS topic that has a subscription filter policy.

B. Create a subscription filter on the log group that has a filter pattern.
C. Set up an Amazon CloudWatch agent operator to manage the trace collection daemon in Amazon EKS.
D. Create an AWS Lambda function to process the logs.
E. Create an Amazon EventBridge rule to invoke the AWS Lambda function on a schedule.
Answer: (SHOW ANSWER)
Requirement Summary:
EKS containers send logs to CloudWatch Logs
Need to process logs in real time
Trigger logic based on a specific error in logs
Evaluate Options:
Option A: SNS topic with filter policy
SNS filter policies work on message attributes, not on CloudWatch Logs subscription filters
Option B: Subscription filter on log group This enables real-time log processing You can create a subscription filter with a pattern matching specific error strings Sends matched logs to a Lambda function or Kinesis Option C: CloudWatch agent operator for trace collection Irrelevant for log processing Used for monitoring and tracing, not real-time log filtering Option D: Lambda function to process logs Once logs match the pattern, Lambda can process and act (e.g., alert, store, analyze) Option E: EventBridge rule on a schedule Not real-time Scheduled EventBridge rules are for cron-like tasks, not log stream processing Subscription filters:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/SubscriptionFilters.html
Real-time log processing with Lambda:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs
/SubscriptionFilters.html#LambdaExample
Logs in EKS to CloudWatch: https://docs.aws.amazon.com/eks/latest/userguide/fargate-logging.html

NEW QUESTION: 59
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: B,C (LEAVE A REPLY)

Requirement Summary:
DynamoDB Provisioned Mode
ProvisionedThroughputExceededException error occurring
App hosted on a small burstable EC2 instance
Option A: Move to a larger EC2 instance
May improve local performance, but does not solve DynamoDB provisioned throughput limits.
Option B: Increase DynamoDB RCUs
Valid: This directly increases the read throughput capacity of the table.
Helps handle more traffic and reduce throughput exceptions.
Option C: Reduce frequency via exponential backoff
Best practice: Using exponential backoff and jitter reduces request pressure and spreads retries out to avoid spiking.
Option D: Increase frequency of retries by reducing delay
Opposite of best practice. Increases load, worsening the issue.
Option E: Change to on-demand mode
Also valid in general, but question is scoped around provisioned capacity.
Since minimizing cost or workload type isn't mentioned, and scaling the existing model is the focus, prefer B and C.
ProvisionedThroughputExceededException:
https://docs.aws.amazon.com/amazondynamodb/latest
/developerguide/HandlingErrors.html
Exponential backoff:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Programming.Errors.html
Capacity modes:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html

NEW QUESTION: 60
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: (SHOW ANSWER)

The UnprocessedKeys element indicates that the BatchGetItem operation did not process all of the requested items in the current response. This can happen if the response size limit is exceeded or if the table's provisioned throughput is exceeded. To handle this situation, the developer should retry the batch operation with exponential backoff and randomized delay to avoid throttling errors and reduce the load on the table. The developer should also use an AWS SDK to make the requests, as the SDKs automatically retry requests that return UnprocessedKeys.

References:

* [BatchGetItem - Amazon DynamoDB]
* [Working with Queries and Scans - Amazon DynamoDB]
* [Best Practices for Handling DynamoDB Throttling Errors]

NEW QUESTION: 61

A company runs a new application on AWS Elastic Beanstalk. The company needs to deploy updates to the application. The updates must not cause any downtime for application users. The deployment must forward a specified percentage of incoming client traffic to a new application version during an evaluation period.

Which deployment type will meet these requirements?

A. Rolling

B. Traffic-splitting

C. In-place

D. Immutable

Answer: B (LEAVE A REPLY)

AWS Elastic Beanstalk supports several deployment policies, and in this case, the requirement is to forward a specific percentage of traffic to the new version without causing downtime. The Traffic-splitting deployment policy is the most appropriate choice.

Traffic-splitting Deployment: This deployment method allows you to gradually shift a specified percentage of incoming traffic from the old environment version to the new one. During the evaluation period, if any issues are detected, the traffic can be redirected back to the old version.

No Downtime: This method ensures no downtime since both versions of the application run concurrently, and traffic is split between them.

Alternatives:

Rolling deployments (Option A): These gradually replace instances but may result in partial downtime if some instances fail during deployment.

In-place deployments (Option C): In-place deployments replace instances without creating new ones, which can lead to downtime.

Immutable deployments (Option D): While this ensures no downtime by creating entirely new instances, it doesn't provide traffic splitting during the evaluation phase.
Elastic Beanstalk Deployment Policies

NEW QUESTION: 62
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: (SHOW ANSWER)
* Why Option B is Correct:A customer-managed AWS Key Management Service (KMS) key allows for encryption at rest and provides the ability to rotate the key on demand. This ensures compliance with security requirements for key management and database encryption.
* RDS integrates natively with AWS KMS, allowing the use of a customer-managed key for encrypting data at rest.
* Key rotation can be managed directly in AWS KMS without needing custom solutions.
* Why Other Options are Incorrect:
* Option A:AWS KMS managed encryption keys (AWS-owned keys) do not support key rotation on demand.
* Option C & D:Storing keys in AWS Secrets Manager with custom rotation is not a recommended approach for database encryption. AWS KMS is designed specifically for secure key management and encryption.
* AWS Documentation References:
* Encrypting Amazon RDS Resources

* AWS Key Management Service (KMS)

NEW QUESTION: 63
A company wants to migrate applications from its on-premises servers to AWS. As a first step, the company is modifying and migrating a non-critical application to a single Amazon EC2 instance. The application will store information in an Amazon S3 bucket. The company needs to follow security best practices when deploying the application on AWS. Which approach should the company take to allow the application to interact with Amazon S3?
A. Create an 1AM user. Attach a policy that provides the necessary access to Amazon S3. Copy the generated access key and secret key. Within the application code, use the access key and secret key along with the AWS SDK to communicate with Amazon S3.
B. Create an 1AM user. Attach the AdministratorAccess policy. Copy the generated access key and secret key. Within the application code, use the access key and secret key along with the AWS SDK to communicate with Amazon S3.
C. Create an 1AM role that has the necessary access to Amazon S3. Attach the role to the EC2 instance.
D. Create an 1AM role that has administrative access to AWS. Attach the role to the EC2 instance.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 64
A company had an Amazon RDS for MySQL DB instance that was named mysql-db. The DB instance was deleted within the past 90 days. A developer needs to find which 1AM user or role deleted the DB instance in the AWS environment. Which solution will provide this information?
A. Retrieve the Amazon CloudWatch log events from the most recent log stream within the rds/mysql-db log group. Inspect the log events.
B. Retrieve the AWS CloudTrail events for the resource mysql-db where the event name is DeleteDBInstance. Inspect each event.
C. Retrieve the AWS Systems Manager deletions inventory Filter the inventory by deletions that have a TypeName value of RDS. Inspect the deletion details.
D. Retrieve the AWS X-Ray trace summaries. Filter by services with the name mysql-db. Inspect the ErrorRootCauses values within each summary.
Answer: B (LEAVE A REPLY)

NEW QUESTION: 65
A company uses AWS Secrets Manager to store API keys for external REST services. The company uses an AWS Lambda function to rotate the API keys on a regular schedule. Due to an error in the Lambda function, the API keys are successfully updated in AWS Secrets Manager but are not updated in the external REST services. Before investigating

the root cause of the issue, the company wants to resume requests to the external REST services as quickly as possible.
Which solution will meet this requirement with the LEAST operational overhead?
A. Manually create a new version of the API keys in AWS Secrets Manager and update the keys in the external REST services.
B. Manually retrieve the new version of the API keys from AWS Secrets Manager and update the keys in the external REST services.
C. Roll back to the last known working version of the API keys in AWS Secrets Manager.
D. Fix and reinvoke the AWS Lambda rotation function to generate a new version of the API keys in AWS Secrets Manager and update the keys in the external REST services.
Answer: C (LEAVE A REPLY)
Comprehensive and Detailed Explanation (250-300 words) From Exact Extract of AWS Developer Documents:
AWS Secrets Manager is designed to securely store, rotate, and manage sensitive information such as API keys, database credentials, and tokens. A core feature of Secrets Manager is secret versioning, which enables recovery from failed or partial rotation events. Each secret version is associated with staging labels, most notably AWSCURRENT and AWSPREVIOUS.
According to AWS documentation, during a successful rotation, Secrets Manager creates a new secret version and assigns it the AWSCURRENT label, while the previous version is retained with the AWSPREVIOUS label. Applications that retrieve secrets typically reference the secret using the AWSCURRENT label. If a rotation fails after updating the secret value but before synchronizing the change with the external system, authentication failures can occur.
AWS explicitly states that in such scenarios, customers can restore service quickly by moving the AWSCURRENT label back to the previous version. This rollback capability allows applications to immediately resume using the last known valid credentials without modifying application code, redeploying resources, or interacting with external systems.
This approach represents the least operational overhead because it is a metadata-only operation within Secrets Manager. No new secrets are created, no Lambda functions are invoked, and no manual updates are required in the external REST services. Additionally, AWS recommends postponing troubleshooting until service availability is restored, which aligns precisely with the requirement in this scenario.
Other options introduce unnecessary complexity, increased risk of human error, and longer recovery times.
Therefore, rolling back to the last working secret version is the fastest, safest, and AWS-recommended solution.

NEW QUESTION: 66

A company's application runs on a fleet of Amazon EC2 instances in a VPC within private subnets that do not have public internet access. The company uses Amazon CloudWatch to monitor the application.
A developer is troubleshooting an issue with the application. Some performance metrics are not being published to CloudWatch. The developer uses EC2 Instance Connect to access an EC2 instance. The developer verifies that a CloudWatch agent is pre-installed and running.
The developer needs to ensure that the performance metrics are published to CloudWatch.
Which solution will meet this requirement in the MOST secure way?
A. Attach the CloudWatchAgentAdminPolicy managed IAM policy to the IAM role that is associated with the EC2 instance profile. Provision a NAT gateway in a public subnet.
B. Add a user data script to install and start up the CloudWatch agent automatically when the EC2 instances are first booted up.
C. Attach the CloudWatchAgentServerPolicy managed IAM policy to the IAM role that is associated with the EC2 instance profile. Provision a VPC interface endpoint for CloudWatch.
D. Attach the CloudWatchReadOnlyAccess managed IAM policy to the IAM role that is associated with the EC2 instance profile. Provision a VPC interface endpoint for CloudWatch.
Answer: C (LEAVE A REPLY)
The EC2 instances are in private subnets with no public internet access, yet the CloudWatch agent must publish metrics to Amazon CloudWatch. To send metrics, the instances need: (1) network connectivity to CloudWatch APIs, and (2) IAM permissions that allow the agent to publish metrics and logs.
The most secure approach is to avoid opening outbound internet access via a NAT gateway (which increases the network attack surface) and instead use AWS PrivateLink through a VPC interface endpoint for CloudWatch. An interface endpoint provides private connectivity from the VPC to CloudWatch without traversing the public internet.
On the permissions side, AWS provides the managed IAM policy CloudWatchAgentServerPolicy, which is intended for the CloudWatch agent running on servers to publish metrics/logs. Attaching this policy to the EC2 instance profile role provides the needed permissions while keeping scope appropriate (unlike an admin policy).
Why the other options are not best:
A uses CloudWatchAgentAdminPolicy (broader permissions than needed) and a NAT gateway (public internet path). That is not "most secure." B is irrelevant here because the agent is already installed and running; the issue is publishing connectivity /permissions.
D grants read-only access, which cannot publish metrics, so it cannot fix the issue.
Therefore, attach CloudWatchAgentServerPolicy to the instance role and create a VPC interface endpoint for CloudWatch.

NEW QUESTION: 67
Users are reporting errors in an application. The application consists of several micro services that are deployed on Amazon Elastic Container Serves (Amazon ECS) with AWS Fargate.
When combination of steps should a developer take to fix the errors? (Select TWO)
A. Deploy AWS X-Ray as a sidecar container to the micro services. Update the task role policy to allow access to me X -Ray API.
B. Deploy AWS X-Ray as a daemon set to the Fargate cluster. Update the service role policy to allow access to the X-Ray API.
C. Instrument the application by using the AWS X-Ray SDK. Update the application to use the Put- XrayTrace API call to communicate with the X-Ray API.
D. Instrument the application by using the AWS X-Ray SDK. Update the application to communicate with the X-Ray daemon.
E. Instrument the ECS task to send the stout and spider- output to Amazon CloudWatch Logs. Update the task role policy to allow the cloudwatch Putlogs action.
Answer: A,E (LEAVE A REPLY)
The combination of steps that the developer should take to fix the errors is to deploy AWS X-Ray as a sidecar container to the microservices and instrument the ECS task to send the stdout and stderr output to Amazon CloudWatch Logs. This way, the developer can use AWS X-Ray to analyze and debug the performance of the microservices and identify any issues or bottlenecks. The developer can also use CloudWatch Logs to monitor and troubleshoot the logs from the ECS task and detect any errors or exceptions. The other options either involve using AWS X-Ray as a daemon set, which is not supported by Fargate, or using the PutTraceSegments API call, which is not necessary when using a sidecar container.
Reference: Using AWS X-Ray with Amazon ECS

NEW QUESTION: 68
A developer is creating an AWS Lambda function that searches for items from an Amazon DynamoDB table that contains customer contact information. The DynamoDB table items have the customer's email_address as the partition key and additional properties such as customer_type, name, and job_title.
The Lambda function runs whenever a user types a new character into the customer_type text input. The developer wants the search to return partial matches of the email_address property for a particular customer_type value. The developer does not want to recreate the DynamoDB table.
What should the developer do to meet these requirements?
A. Add a global secondary index (GSI) to the DynamoDB table with customer_type as the partition key and email_address as the sort key. Perform a query operation on the GSI by using the begins_with key condition expression with the email_address property.

B. Add a global secondary index (GSI) to the DynamoDB table with email_address as the partition key and customer_type as the sort key. Perform a query operation on the GSI by using the begins_with key condition expression with the email_address property.
C. Add a local secondary index (LSI) to the DynamoDB table with customer_type as the partition key and email_address as the sort key. Perform a query operation on the LSI by using the begins_with key condition expression with the email_address property.
D. Add a local secondary index (LSI) to the DynamoDB table with job_title as the partition key and email_address as the sort key. Perform a query operation on the LSI by using the begins_with key condition expression with the email_address property.
Answer: A (LEAVE A REPLY)
The requirement is to search for partial matches of email_address but scoped to a specific customer_type. In DynamoDB, efficient partial matching is performed with a Query on a sort key using the begins_with() condition. However, Query requires that the partition key be specified exactly, so we need an index where customer_type can be used as the partition key, and email_address can be used as the sort key for prefix matching.
A Global Secondary Index (GSI) can be added to an existing table without recreating it, and it can use a different partition key and sort key from the base table. Option A correctly proposes a GSI with:
Partition key: customer_type
Sort key: email_address
Then the Lambda can run a Query on the GSI like: customer_type = :ct AND begins_with(email_address, :
prefix) which returns emails that start with the typed prefix for that customer type.
Option B is wrong because it keeps email_address as the partition key. That would require an exact email value (not a prefix) to Query, and you cannot do begins_with on a partition key; begins_with is for sort keys.
Options C and D are invalid because Local Secondary Indexes (LSIs) must share the same partition key as the base table (here, email_address). You cannot create an LSI with customer_type or job_title as the partition key when the table partition key is email_address.
Therefore, adding a GSI (customer_type PK, email_address SK) and querying it with begins_with is the correct solution.

NEW QUESTION: 69
A developer is building an application that uses AWS API Gateway APIs. AWS Lambda function, and AWS Dynamic DB tables. The developer uses the AWS Serverless Application Model (AWS SAM) to build and run serverless applications on AWS. Each time the developer pushes of changes for only to the Lambda functions, all the artifacts in the application are rebuilt.
The developer wants to implement AWS SAM Accelerate by running a command to only redeploy the Lambda functions that have changed.

Which command will meet these requirements?

A. sam deploy -force-upload

B. sam deploy -no-execute-changeset

C. sam package

D. sam sync -watch

Answer: D (LEAVE A REPLY)

The command that will meet the requirements is sam sync -watch. This command enables AWS SAM Accelerate mode, which allows the developer to only redeploy the Lambda functions that have changed. The - watch flag enables file watching, which automatically detects changes in the source code and triggers a redeployment. The other commands either do not enable AWS SAM Accelerate mode, or do not redeploy the Lambda functions automatically.

Reference: AWS SAM Accelerate

NEW QUESTION: 70

A developer is running an application on an Amazon EC2 instance. When the application attempts to read from an Amazon S3 bucket, the request fails. The developer determines that the IAM role associated with the EC2 instance is missing the required Amazon S3 read permissions.

The developer must grant the application access to read from the S3 bucket with the LEAST application disruption.

Which solution will meet this requirement?

A. Add the permission to the IAM role. Terminate the EC2 instance and launch a new instance.

B. Add the permission to the IAM role so that the change takes effect automatically.

C. Add the permission to the IAM role. Hibernate and restart the EC2 instance.

D. Add the permission to the S3 bucket and restart the EC2 instance.

Answer: B (LEAVE A REPLY)

AWS IAM role permissions are evaluated dynamically. According to AWS documentation, when an IAM role attached to an EC2 instance is updated, the updated permissions are available to applications running on that instance without requiring a restart. The EC2 instance retrieves temporary credentials through the instance metadata service, and those credentials are refreshed automatically.

In this scenario, the least disruptive approach is to update the IAM role policy to include the required s3:

GetObject permission. Once the permission is added, the application can immediately retry access to the S3 bucket without restarting, stopping, or terminating the EC2 instance.

Terminating or restarting the instance (Options A and C) introduces unnecessary downtime and operational overhead. Modifying the S3 bucket policy (Option D) is unnecessary because the problem lies with missing role permissions, not bucket access configuration.

AWS best practices strongly recommend using IAM roles for EC2 and updating permissions dynamically to avoid service interruption. Therefore, simply adding the required permission to the role is the correct and most efficient solution.

NEW QUESTION: 71
A developer is storing JSON files in an Amazon S3 bucket. The developer wants to securely share an object with a specific group of people.
How can the developer securely provide temporary access to the objects that are stored in the S3 bucket?
A. Set object retention on the files. Use the AWS SDK to restore the object before subsequent requests.
Provide the bucket's S3 URL.
B. Use the AWS SDK to generate a presigned URL. Provide the presigned URL.
C. Set a bucket policy that restricts access after a period of time. Provide the bucket's S3 URL.
D. Configure static web hosting on the S3 bucket. Provide the bucket's web URL.
Answer: (SHOW ANSWER)
To securely grant temporary access to a specific S3 object, the best option is to use an Amazon S3 presigned URL. A presigned URL is created by an IAM principal (user or role) that already has permission to access the object. The URL includes a signature and an expiration time, allowing anyone who has the URL to perform the specified operation (typically GET to download, or PUT to upload) until the URL expires. This meets the requirement for temporary access without changing the bucket to be publicly accessible.
Presigned URLs are especially useful when the developer needs to share access with people who do not have AWS credentials or are not part of the same AWS account. The developer can set an expiration that matches the business need (minutes to hours, etc.). After expiration, the URL no longer works, which reduces security exposure compared with long-lived access keys or permanently permissive bucket policies.
Option C (bucket policy with time restriction) can enforce time-based access, but it is not a practical way to securely share with a "specific group of people" unless those people authenticate with identifiable principals (IAM roles/users, federated identities) and you scope the policy to those principals. Simply providing the bucket S3 URL would not grant access unless the recipients also have valid permissions. Option D (static website hosting) is generally for public web content and does not inherently provide secure, temporary, per-object access control. Option A is unrelated to access sharing; object retention and restore operations are for retention/compliance and archival retrieval, not temporary permission grants.
Therefore, generating and sharing a presigned URL (B) is the standard, secure mechanism for time-limited access to S3 objects.

NEW QUESTION: 72

A developer is migrating an application to Amazon Elastic Kubernetes Service (Amazon EKS). The developer migrates the application to Amazon Elastic Container Registry (Amazon ECR) with an EKS cluster.
As part of the application migration to a new backend, the developer creates a new AWS account. The developer makes configuration changes to the application to point the application to the new AWS account and to use new backend resources. The developer successfully tests the changes within the application by deploying the pipeline.
The Docker image build and the pipeline deployment are successful, but the application is still connecting to the old backend. The developer finds that the application's configuration is still referencing the original EKS cluster and not referencing the new backend resources.
Which reason can explain why the application is not connecting to the new resources?
A. The developer did not successfully create the new AWS account.
B. The developer added a new tag to the Docker image.
C. The developer did not update the Docker image tag to a new version.
D. The developer pushed the changes to a new Docker image tag.
Answer: (SHOW ANSWER)
The correct answer is C. The developer did not update the Docker image tag to a new version.
C). The developer did not update the Docker image tag to a new version. This is correct. When deploying an application to Amazon EKS, the developer needs to specify the Docker image tag that contains the application code and configuration. If the developer does not update the Docker image tag to a new version after making changes to the application, the EKS cluster will continue to use the old Docker image tag that references the original backend resources. To fix this issue, the developer should update the Docker image tag to a new version and redeploy the application to the EKS cluster.
A). The developer did not successfully create the new AWS account. This is incorrect. The creation of a new AWS account is not related to the application's connection to the backend resources. The developer can use any AWS account to host the EKS cluster and the backend resources, as long as they have the proper permissions and configurations.
B). The developer added a new tag to the Docker image. This is incorrect. Adding a new tag to the Docker image is not enough to deploy the changes to the application. The developer also needs to update the Docker image tag in the EKS cluster configuration, so that the EKS cluster can pull and run the new Docker image.
D). The developer pushed the changes to a new Docker image tag. This is incorrect. Pushing the changes to a new Docker image tag is not enough to deploy the changes to the application. The developer also needs to update the Docker image tag in the EKS cluster configuration, so that the EKS cluster can pull and run the new Docker image.
References:
1: Amazon EKS User Guide, "Deploying applications to your Amazon EKS cluster", https://docs.aws.amazon.
com/eks/latest/userguide/deploying-applications.html

2: Amazon ECR User Guide, "Pushing an image",
https://docs.aws.amazon.com/AmazonECR/latest/userguide
/docker-push-ecr-image.html
3: Amazon EKS User Guide, "Updating an Amazon EKS cluster",
https://docs.aws.amazon.com/eks/latest
/userguide/update-cluster.html

NEW QUESTION: 73
An online sales company is developing a serverless application that runs on AWS. The application uses an AWS Lambda function that calculates order success rates and stores the data in an Amazon DynamoDB table.
A developer wants an efficient way to invoke the Lambda function every 15 minutes.
Which solution will meet this requirement with the LEAST development effort?
A. Create an Amazon EventBridge rule that has a rate expression that will run the rule every 15 minutes.
Add the Lambda function as the target of the EventBridge rule.
B. Create an AWS Systems Manager document that has a script that will invoke the Lambda function on Amazon EC2. Use a Systems Manager Run Command task to run the shell script every 15 minutes.
C. Create an AWS Step Functions state machine. Configure the state machine to invoke the Lambda function execution role at a specified interval by using a Wait state. Set the interval to 15 minutes.
D. Provision a small Amazon EC2 instance. Set up a cron job that invokes the Lambda function every 15 minutes.
Answer: (SHOW ANSWER)
The best solution for this requirement is option A. Creating an Amazon EventBridge rule that has a rate expression that will run the rule every 15 minutes and adding the Lambda function as the target of the EventBridge rule is the most efficient way to invoke the Lambda function periodically. This solution does not require any additional resources or development effort, and it leverages the built-in scheduling capabilities of EventBridge1.

NEW QUESTION: 74
A company uses an AWS Lambda function to perform natural language processing (NLP) tasks. The company has attached a Lambda layer to the function. The Lambda layer contain scientific libraries that the function uses during processing.
The company added a large, pre-trained text-classification model to the Lambda layer. The addition increased the size of the Lambda layer to 8.7 GB. After the addition and a recent deployment, the Lambda function returned a RequestEntityTooLargeException error.
The company needs to update the Lambda function with a high-performing and portable solution to decrease the initialization time for the function.
Which solution will meet these requirements?

A. Store the large pre-trained model in an Amazon S3 bucket. Use the AWS SDK to access the model.
B. Create an Amazon EFS file system to store the large pre-trained model. Mount the file system to an Amazon EC2 instance. Configure the Lambda function to use the EFS file system.
C. Split the components of the Lambda layer into five new Lambda layers. Zip the new layers, and attach the layers to the Lambda function. Update the function code to use the new layers.
D. Create a Docker container that includes the scientific libraries and the pre-trained model. Update the Lambda function to use the container image.
Answer: D (LEAVE A REPLY)
Requirement Summary:
NLP Lambda function with a large pre-trained model
Lambda layer became 8.7 GB # Exceeds AWS limits
Function returns RequestEntityTooLargeException
Need: High-performing, portable, low initialization time
Important AWS Limits:
Lambda Layers size limit (combined across all layers): 250 MB (unzipped) Deployment package size (unzipped): 250 MB Lambda container image support allows up to 10 GB image size Evaluate Options:
A: Store model in S3 and load during execution
Leads to cold start latency every time
Model loading from S3 is slower and not suitable for real-time NLP
Not optimal for performance
B: Use EFS mounted to Lambda
Valid for large models, but adds latency during cold start as model loads from EFS Requires EFS setup, VPC, and has added network I/O overhead Still slower than bundling in container image C: Split into five Lambda layers Still violates the total layer size limit of 250 MB (unzipped) You cannot exceed that even with multiple layers D: Use Docker container image Allows bundling up to 10 GB of dependencies and models High portability and performance Avoids latency of downloading models at runtime Ideal for scientific/NLP models Lambda container image support:
https://docs.aws.amazon.com/lambda/latest/dg/images-create.html Lambda limits:
https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html Using large models with Lambda: https://aws.amazon.com/blogs/machine-learning/deploying-large-machine- learning-models-on-aws-lambda-with-container-images/

NEW QUESTION: 75
A company runs a payment application on Amazon EC2 instances behind an Application Load Balance The EC2 instances run in an Auto Scaling group across multiple Availability Zones The application needs to retrieve application secrets during the application startup

and export the secrets as environment variables These secrets must be encrypted at rest and need to be rotated every month.
Which solution will meet these requirements with the LEAST development effort?
A. Save the secrets in a text file and store the text file in Amazon S3 Provision a customer managed key Use the key for secret encryption in Amazon S3 Read the contents of the text file and read the export as environment variables Configure S3 Object Lambda to rotate the text file every month
B. Save the secrets as strings in AWS Systems Manager Parameter Store and use the default AWS Key Management Service (AWS KMS) key Configure an Amazon EC2 user data script to retrieve the secrets during the startup and export as environment variables Configure an AWS Lambda function to rotate the secrets in Parameter Store every month.
C. Save the secrets as base64 encoded environment variables in the application properties. Retrieve the secrets during the application startup. Reference the secrets in the application code. Write a script to rotate the secrets saved as environment variables.
D. Store the secrets in AWS Secrets Manager Provision a new customer master key Use the key to encrypt the secrets Enable automatic rotation Configure an Amazon EC2 user data script to programmatically retrieve the secrets during the startup and export as environment variables
Answer: (SHOW ANSWER)
* AWS Secrets Manager: Built for managing secrets, providing encryption, automatic rotation, and access control.
* Customer Master Key (CMK): Provides an extra layer of control over encryption through AWS KMS.
* Automatic Rotation: Enhances security by regularly changing the secret.
* User Data Script: Allows secrets retrieval at instance startup and sets them as environment variables for seamless use within the application.
References:
AWS Secrets Manager Documentation: https://docs.aws.amazon.com/secretsmanager/
AWS KMS Documentation: https://docs.aws.amazon.com/kms/ User Data for EC2 Instances: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html

NEW QUESTION: 76
A company has an application that is based on Amazon EC2. The company provides API access to the application through Amazon API Gateway and uses Amazon DynamoDB to store the application's data. A developer is investigating performance issues that are affecting the application. During peak usage, the application is overwhelmed by a large number of identical data read requests that come through APIs. What is the MOST operationally efficient way for the developer to improve the application's performance?
A. Use DynamoDB Accelerator (DAX) to cache database responses.
B. Configure Amazon EC2 Auto Scaling policies to meet fluctuating demand.
C. Enable API Gateway caching to cache API responses.

D. Use Amazon ElastiCache to cache application responses.

Answer: A (LEAVE A REPLY)

DynamoDB Accelerator (DAX) provides a managed caching layer specifically optimized for DynamoDB, reducing latency for repeated read requests.

* Why Option A is Correct:

* Purpose-Built: DAX is designed for DynamoDB, enabling sub-millisecond response times for frequently accessed items.

* Operational Efficiency: No need for additional application-level caching logic.

* Why Not Other Options:

* Option B: Auto Scaling increases capacity but does not address repetitive reads.

* Option C: API Gateway caching helps reduce request processing time but does not optimize DynamoDB reads.

* Option D: ElastiCache is a general-purpose cache, adding unnecessary complexity for DynamoDB use cases.

DynamoDB Accelerator (DAX)

NEW QUESTION: 77

A developer is creating an AWS Lambda function that consumes messages from an Amazon Simple Queue Service (Amazon SQS) standard queue. The developer notices that the Lambda function processes some messages multiple times.

How should developer resolve this issue MOST cost-effectively?

A. Change the Amazon SQS standard queue to an Amazon SQS FIFO queue by using the Amazon SQS message deduplication ID.

B. Set up a dead-letter queue.

C. Set the maximum concurrency limit of the AWS Lambda function to 1

D. Change the message processing to use Amazon Kinesis Data Streams instead of Amazon SQS.

Answer: (SHOW ANSWER)

Amazon Simple Queue Service (Amazon SQS) is a fully managed queue service that allows you to de-couple and scale for applications1. Amazon SQS offers two types of queues: Standard and FIFO (First In First Out) queues1. The FIFO queue uses the messageDeduplicationId property to treat messages with the same value as duplicate2.

Therefore, changing the Amazon SQS standard queue to an Amazon SQS FIFO queue using the Amazon SQS message deduplication ID can help resolve the issue of the Lambda function processing some messages multiple times. Therefore, option A is correct.

NEW QUESTION: 78
A developer is creating an Amazon DynamoDB table by using the AWS CLI The DynamoDB table must use server-side encryption with an AWS owned encryption key How should the developer create the DynamoDB table to meet these requirements?
A. Create an AWS Key Management Service (AWS KMS) customer managed key. Provide the key's Amazon Resource Name (ARN) in the KMSMasterKeyId parameter during creation of the DynamoDB table
B. Create an AWS Key Management Service (AWS KMS) AWS managed key Provide the key's Amazon Resource Name (ARN) in the KMSMasterKeyId parameter during creation of the DynamoDB table
C. Create an AWS owned key Provide the key's Amazon Resource Name (ARN) in the KMSMasterKeyId parameter during creation of the DynamoDB table.
D. Create the DynamoDB table with the default encryption options
Answer: (SHOW ANSWER)
Default SSE in DynamoDB: DynamoDB tables are encrypted at rest by default using an AWS owned key (SSE-S3).
No Additional Action Needed: Creating a table without explicitly specifying a KMS key will use this default encryption.
References:
DynamoDB Server-Side Encryption:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide
/Encryption

NEW QUESTION: 79
A developer is creating an application that must transfer expired items from Amazon DynamoDB to Amazon S3. The developer sets up the DynamoDB table to automatically delete items after a specific TTL. The application must process the items in DynamoDB and then must store the expired items in Amazon S3. The entire process, including item processing and storage in Amazon S3, will take 5 minutes.
Which solution will meet these requirements with the LEAST operational overhead?
A. Deploy a custom application on an Amazon Elastic Container Service (Amazon ECS) cluster on Amazon EC2 instances. Program the custom application to process the items and to store the expired items in Amazon S3.
B. Create an Amazon EventBridge rule to invoke an AWS Lambda function. Program the Lambda function to process The items and to store the expired items in Amazon S3.
C. Configure DynamoDB Accelerator (DAX) lo query for expired items based on the TTL Save the results to Amazon S3.

D. Configure DynamoDB Streams to invoke an AWS Lambda function. Program the Lambda function to process the items and to store the expired items in Amazon S3.
Answer: D (LEAVE A REPLY)

NEW QUESTION: 80
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: (SHOW ANSWER)
This solution will meet the requirements by using AWS Config to monitor and evaluate whether Secrets Manager secrets are unused or have been deleted, based on specified time periods. The secrets manager-secret- unused managed rule is a predefined rule that checks whether Secrets Manager secrets have been rotated within a specified number of days or have been deleted within a specified number of days after last accessed date. The Amazon EventBridge rule will trigger a notification when the AWS Config managed rule is met, alerting the developer about unused secrets that can be removed without causing application downtime.
Option A is not optimal because it will use AWS CloudTrail log file delivery to an Amazon S3 bucket, which will incur additional costs and complexity for storing and analyzing log files that may not contain relevant information about secret usage. Option C is not optimal because it will deactivate the application secrets and monitor the application error logs temporarily, which will cause application downtime and potential data loss.
Option D is not optimal because it will use AWS X-Ray to trace secret usage, which will introduce additional overhead and latency for instrumenting and sampling requests that may not be related to secret usage.
References: [AWS Config Managed Rules], [Amazon EventBridge]

NEW QUESTION: 81

A developer is troubleshooting a three-tier application, which is deployed on Amazon EC2 instances. There is a connectivity problem between the application servers and the database servers.
Which AWS services or tools should be used to identify the faulty component? (Select TWO.)
A. AWS Trusted Advisor
B. Amazon VPC Flow Logs
C. AWS CloudTrail
D. Network access control lists
E. AWS Config rules
Answer: B,D (LEAVE A REPLY)

NEW QUESTION: 82
A developer needs to use Amazon DynamoDB to store customer orders. The developer's company requires all customer data to be encrypted at rest with a key that the company generates.
What should the developer do to meet these requirements?
A. Create the DynamoDB table with encryption set to None. Code the application to use the key to decrypt the data when the application reads from the table. Code the application to use the key to encrypt the data when the application writes to the table.
B. Store the key by using AW5 KMS. Choose an AVVS KMS customer managed key during creation of the DynamoDB table. Provide the Amazon Resource Name (ARN) of the AWS KMS key.
C. Store the key by using AWS KMS. Create the DynamoDB table with default encryption. Include the kms:Encrypt parameter with the Amazon Resource Name (ARN) of the AWS KMS key when using the DynamoDB SDK.
D. Store the key by using AWS KMS. Choose an AWS KMS AWS managed key during creation of the DynamoDB table. Provide the Amazon Resource Name (ARN) of the AWS KMS key.
Answer: B (LEAVE A REPLY)
Requirement Summary:
* Store customer orders in DynamoDB
* Must encrypt data at rest
* Company wants to use a key it generates (i.e., customer managed key)
Evaluate Options:
A). Set encryption to None, manually encrypt/decrypt in code
* Overhead and error-prone
* Also non-compliant with AWS encryption best practices
B). Use customer managed KMS key
* Exactly meets the requirement: customer generates and controls the key
* During table creation, you can specify a KMS CMK ARN

C). Default encryption + kms:Encrypt in SDK
* Misunderstanding: DynamoDB handles encryption automatically
* You don't need to call kms:Encrypt manually in SDK
D). Use AWS managed key
* Does not meet the requirement of using custom company-generated key
* DynamoDB encryption:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide
/EncryptionAtRest.html
* KMS customer managed keys:
https://docs.aws.amazon.com/kms/latest/developerguide/concepts.
html#customer-cmk

NEW QUESTION: 83
A developer is building an application that gives users the ability to view bank account from multiple sources in a single dashboard. The developer has automated the process to retrieve API credentials for these sources.
The process invokes an AWS Lambda function that is associated with an AWS CloudFormation cotton resource.
The developer wants a solution that will store the API credentials with minimal operational overhead.
When solution will meet these requirements?
A. Add an AWS Secrets Manager GenerateSecretString resource to the CloudFormation template. Set the value to reference new credentials to the Cloudformation resource.
B. Use the AWS SDK ssm PutParameter operation in the Lambda function from the existing, custom resource to store the credentials as a parameter. Set the parameter value to reference the new credentials. Set ma parameter type to SecureString.
C. Add an AWS Systems Manager Parameter Store resource to the CloudFormation template. Set the CloudFormation resource value to reference the new credentials Set the resource NoEcho attribute to true.
D. Use the AWS SDK ssm PutParameter operation in the Lambda function from the existing custom resources to store the credentials as a parameter. Set the parameter value to reference the new credentials. Set the parameter NoEcho attribute to true.
Answer: B (LEAVE A REPLY)
The solution that will meet the requirements is to use the AWS SDK ssm PutParameter operation in the Lambda function from the existing custom resource to store the credentials as a parameter. Set the parameter value to reference the new credentials. Set the parameter type to SecureString. This way, the developer can store the API credentials with minimal operational overhead, as AWS Systems Manager Parameter Store provides secure and scalable storage for configuration data. The SecureString parameter type encrypts the parameter value with AWS Key Management Service (AWS KMS). The other options either involve adding additional resources to the CloudFormation template, which

increases complexity and cost, or do not encrypt the parameter value, which reduces security.
Reference: Creating Systems Manager parameters

NEW QUESTION: 84
A developer is writing unit tests tor a new application that will be deployed on AWS. The developer wants to validate all pull requests with unit tests and merge the code with the main branch only when all tests pass The developer stores the code in AWS CodeCommit and sets up AWS CodeBuild to run the unit tests. The developer creates an AWS Lambda function to start the CodeBuild task. The developer needs to identify the CodeCommit events in an Amazon EventBridge event that can invoke the Lambda function when a pull request is created or updated.
Which CodeCommit event will meet these requirements?
A. Option A
B. Option B
C. Option C
D. Option D
Answer: C (LEAVE A REPLY)
https://aws.amazon.com/blogs/devops/automated-code-review-on-pull-requests-using-aws-codecommit-and- aws-codebuild/

NEW QUESTION: 85
A developer creates an AWS CloudFormation template that defines one AWS Lambda function, one Amazon S3 bucket, and one IAM role. The developer wants to deploy multiple stacks from the same template across different environments. Each resource must have a unique name per environment.
Which combination of solutions will meet this requirement? (Select TWO.)
A. Create a parameter for the environment name.
B. Create a condition for the environment name.
C. Create a rule for the environment name.
D. Define resource names by using Fn::Sub and !Ref with the environment name parameter.
E. Define resource names by using Fn::GetAtt and !Ref with the environment name parameter.
Answer: A,D (LEAVE A REPLY)
AWS CloudFormation parameters allow templates to be reused across multiple environments by injecting environment-specific values at deployment time. AWS documentation recommends parameters for environment names such as dev, test, and prod.
To ensure unique resource names, CloudFormation intrinsic functions can dynamically construct names. Fn::

Sub allows string interpolation, and !Ref retrieves the parameter value. Together, they enable resource names like myapp-dev-bucket or myapp-prod-role.
Conditions (Option B) control whether resources are created but do not generate unique names. Rules (Option C) validate parameters but do not affect naming. Fn::GetAtt (Option E) retrieves resource attributes and is not used to build names.
Therefore, defining an environment name parameter and using Fn::Sub with !Ref is the correct and AWS- recommended approach.

NEW QUESTION: 86
A company runs a batch processing application by using AWS Lambda functions and Amazon API Gateway APIs with deployment stages for development, user acceptance testing and production A development team needs to configure the APIs in the deployment stages to connect to third-party service endpoints.
Which solution will meet this requirement?
A. Store the third-party service endpoints in Lambda layers that correspond to the stage
B. Store the third-party service endpoints in API Gateway stage variables that correspond to the stage
C. Encode the third-party service endpoints as query parameters in the API Gateway request URL.
D. Store the third-party service endpoint for each environment in AWS AppConfig
Answer: (SHOW ANSWER)
API Gateway Stage Variables: These are designed for configuring dynamic values for your APIs in different deployment stages (dev, test, prod). Here's how to use them for third-party endpoints:
In the API Gateway console, access the "Stages" section of your API.
For each stage, create a stage variable named something like thirdPartyEndpoint.
Set the value of this variable to the actual endpoint URL for that specific environment.
When configuring API requests within your API Gateway method, reference this endpoint using ${stageVariables.thirdPartyEndpoint}.
Why Stage Variables Excel Here:
Environment Isolation: This approach keeps the endpoint configuration specific to each deployment stage, ensuring the right endpoints are used during development, testing, and production cycles.
Ease of Management: You manage the endpoints directly through the API Gateway console without additional infrastructure.
References:
Amazon API Gateway Stage Variables:
https://docs.aws.amazon.com/apigateway/latest/developerguide/stage- variables.html

NEW QUESTION: 87

A developer is writing an application that will provide data files to an external company. The external company needs to verify that the data is not modified in transit.
How can the developer use AWS KMS to prove the integrity of the transferred data?
A. Encrypt the data by using a symmetric key. Provide the key to the external company.
B. Sign the data by using a symmetric key. Provide the key to the external company.
C. Sign the data by using the private key of an asymmetric key pair. Provide the public key to the external company.
D. Sign the data by using the public key of an asymmetric key pair. Provide the private key to the external company.
Answer: (SHOW ANSWER)
To prove integrity (and authenticity) of data in transit, the correct cryptographic primitive is a digital signature. AWS KMS supports signing and verification operations using asymmetric KMS keys designed for signing (for example, RSA_SIGN_PSS, RSA_SIGN_PKCS1, or ECC_NIST_P256 key specs). The developer signs the data (or more commonly, a hash of the data) using the private key, and the external recipient verifies the signature using the corresponding public key. If the data is modified in transit, signature verification fails, proving the content was changed.
Option C exactly describes this model: sign with the private key, share the public key. The private key remains protected (ideally never leaving KMS). The public key can be distributed safely because it cannot be used to forge signatures.
Option A (encryption) provides confidentiality, not integrity proof to a third party, and sharing a symmetric encryption key is insecure and breaks key management principles.
Option B is incorrect because symmetric keys do not provide non-repudiation and generally require the verifier to possess the same secret key, which would allow the verifier to forge signatures too. While HMAC can validate integrity between trusted parties, it does not meet the typical "prove integrity" requirement to an external party without sharing a secret.
Option D is backward and insecure: you never share a private key.
Therefore, use an asymmetric KMS signing key to sign with the private key and provide the public key for verification.

NEW QUESTION: 88
A company runs an application on AWS The application uses an AWS Lambda function that is configured with an Amazon Simple Queue Service (Amazon SQS) queue called high priority queue as the event source A developer is updating the Lambda function with another SQS queue called low priority queue as the event source The Lambda function must always read up to 10 simultaneous messages from the high priority queue before processing messages from low priority queue. The Lambda function must be limited to 100 simultaneous invocations.
Which solution will meet these requirements'?

A. Set the event source mapping batch size to 10 for the high priority queue and to 90 for the low priority queue
B. Set the delivery delay to 0 seconds for the high priority queue and to 10 seconds for the low priority queue
C. Set the event source mapping maximum concurrency to 10 for the high priority queue and to 90 for the low priority queue
D. Set the event source mapping batch window to 10 for the high priority queue and to 90 for the low priority queue
Answer: (SHOW ANSWER)
Lambda Concurrency: The 'maximum concurrency' setting in event source mappings controls the maximum number of simultaneous invocations Lambda allows for that specific source.
Prioritizing Queues: Setting a lower maximum concurrency for the 'high priority queue' ensures it's processed first while allowing more concurrent invocations from the 'low priority queue'.
Batching: Batch size settings affect the number of messages Lambda retrieves from a queue per invocation, which is less relevant to the prioritization requirement.
References:
Lambda Event Source Mappings:
https://docs.aws.amazon.com/lambda/latest/dg/invocation- eventsourcemapping.html
Lambda Concurrency: https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html

NEW QUESTION: 89
A development team wants to run their container workloads on Amazon ECS. Each application container needs to share data with another container to collect logs and metrics.
What should the development team do to meet these requirements?
A. Create two pod specifications. Make one to include the application container and the other to include the other container. Link the two pods together.
B. Create two task definitions. Make one to include the application container and the other to include the other container. Mount a shared volume between the two tasks.
C. Create one task definition. Specify both containers in the definition. Mount a shared volume between those two containers.
D. Create a single pod specification. Include both containers in the specification. Mount a persistent volume to both containers.
Answer: C (LEAVE A REPLY)
On Amazon ECS, the fundamental scheduling unit is a task, which is launched from a task definition. A single task definition can include multiple containers that are intended to run together (often called a

"sidecar" pattern). When two containers must share data locally-such as an application container writing log files and a separate log/metrics collector container reading those files-the recommended approach is to place both containers in the same task definition and use a shared task volume.
ECS supports defining volumes at the task level (for example, a Docker volume or bind mount depending on the launch type and configuration), and then mounting that volume into multiple containers inside the same task. Because the containers are co-located by ECS scheduling (same task, same underlying host or Fargate environment), they can safely share the mounted directory for file-based handoff. This design keeps the containers tightly coupled for lifecycle management (start/stop together) and avoids cross-task coordination complexity.
Options A and D refer to "pod specifications," which are Kubernetes concepts, not ECS-native constructs (unless using ECS with EKS/Kubernetes, which is not stated). Option B is not feasible because ECS does not mount a single shared local volume across two separate tasks in the general case; separate tasks can be placed on different hosts, and task-scoped volumes are not shared between tasks by default. If cross-task sharing were required, you would typically use a network filesystem (like EFS) or an external log pipeline (CloudWatch Logs, FireLens, etc.), but the requirement explicitly says each application container needs to share data with another container, which aligns with the sidecar-in-same-task pattern.
Therefore, the correct solution is C: define both containers in one ECS task definition and mount a shared volume into both containers.

NEW QUESTION: 90
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 (LEAVE A REPLY)
To retrieve a secret's value from AWS Secrets Manager, the calling principal must be authorized to call secretsmanager:GetSecretValue on the specified secret. This API returns the secret material (for example, SecretString or SecretBinary). Passing secret IDs through environment variables only tells the application which secret to request; it does not grant permission to access it.

Because the secret is encrypted with a customer managed AWS KMS key, the caller must also be allowed to use that key for decryption. Secrets Manager stores secret values encrypted at rest, and when a caller requests the secret value, KMS is used to decrypt the stored ciphertext under the configured CMK. Therefore, the principal needs kms:Decrypt permission on the CMK (and the CMK key policy must allow the principal, directly or via grants/conditions). Without kms:Decrypt, the GetSecretValue call will fail because Secrets Manager cannot return plaintext secret material.
secretsmanager:DescribeSecret (B) can be useful for reading metadata such as rotation configuration or tags, but it is not required to retrieve the secret value itself.
secretsmanager:ListSecrets (C) is for discovery
/enumeration and is not required when the application already knows the secret ID.
kms:Encrypt (E) is not needed for reading a secret value; encryption permissions are relevant for write/update operations or client- side encryption flows, not for decrypting stored secrets.
Therefore, the required combination is A (secretsmanager:GetSecretValue) and D (kms:Decrypt) to successfully retrieve secret values encrypted with a customer managed KMS key.

NEW QUESTION: 91
A developer is working on an application that handles 10 MB documents that contain highly sensitive data.
The application will use AWS KMS to perform client-side encryption.
What steps must be followed?
A. Invoke the Encrypt API, passing the plaintext data that must be encrypted, then reference the customer managed key ARN in the KeyId parameter.
B. Invoke the GenerateRandom API to get a data encryption key, then use the data encryption key to encrypt the data.
C. Invoke the GenerateDataKey API to retrieve the encrypted version of the data encryption key to encrypt the data.
D. Invoke the GenerateDataKey API to retrieve the plaintext version of the data encryption key to encrypt the data.
Answer: D (LEAVE A REPLY)
For client-side encryption with AWS KMS, the documented pattern is envelope encryption: you use KMS to generate and protect a data encryption key (DEK), and you use that DEK locally to encrypt the actual data. This is the correct approach for large payloads (such as 10 MB documents), because the KMS Encrypt API is intended for encrypting small amounts of data (KMS is not designed to directly encrypt multi- megabyte application payloads).
With envelope encryption, the application calls GenerateDataKey on AWS KMS, specifying the KMS key (customer managed key) to use. KMS returns two versions of the DEK in the response:

* a plaintext DEK (usable immediately by the application for local cryptographic operations), and
* a ciphertext (encrypted) DEK that is encrypted under the specified KMS key.
The application then uses the plaintext DEK to encrypt the 10 MB document on the client side (for example, using an authenticated encryption mode such as AES-GCM via a standard crypto library). After encryption completes, the application must not persist the plaintext DEK; instead, it stores the encrypted document alongside the ciphertext DEK. Later, to decrypt, the application sends the ciphertext DEK to KMS using Decrypt to recover the plaintext DEK, and then decrypts the document locally.
Therefore, option D is correct because it explicitly uses GenerateDataKey and the plaintext DEK to encrypt the data, which is the required envelope-encryption flow. Options A, B, and C do not follow the correct KMS envelope-encryption process for large client-side payloads.

NEW QUESTION: 92
A developer is creating an AWS Lambda function that will connect to an Amazon RDS for MySQL instance.
The developer wants to store the database credentials. The database credentials need to be encrypted and the database password needs to be automatically rotated.
Which solution will meet these requirements?
A. Store the database credentials as environment variables for the Lambda function. Set the environment variables to rotate automatically.
B. Store the database credentials in AWS Systems Manager Parameter Store as secure string parameters.
Set up managed rotation on the parameters.
C. Store the database credentials in AWS Secrets Manager. Set up managed rotation on the database credentials.
D. Store the database credentials in the X-Amz-Security-Token parameter. Set up managed rotation on the parameter.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 93

A developer wants to use an AWS AppSync API to invoke AWS Lambda functions to return data. Some of the Lambda functions perform long-running processes. The AWS AppSync API needs to return responses immediately.
Which solution will meet these requirements with the LEAST operational overhead?
A. Configure the Lambda functions to be AWS AppSync data sources. Use Event mode for asynchronous Lambda invocation.
B. Increase the timeout setting for the Lambda functions to accommodate longer processing times.
C. Set up an Amazon SQS queue. Configure AWS AppSync to send messages to the SQS queue.
Configure a Lambda function event source mapping to poll the queue.
D. Enable caching, and increase the duration of the AWS AppSync cache TTL.
Answer: (SHOW ANSWER)
Step-by-Step Breakdown:
Requirement Summary:
AWS AppSync API needs to invoke Lambda functions
Some Lambda functions are long-running
AppSync should return immediately, minimizing operational overhead
**Option A: AppSync + Lambda as data source using Event Mode
Correct: AWS AppSync supports asynchronous (event) invocation of Lambda data sources using Event Mode.
Event Mode means:
AppSync invokes Lambda asynchronously
Immediately returns a response to the client (typically a predefined payload or null) Ideal for fire-and-forget workloads or when the response is not immediately needed Option B: Increase Lambda timeout Incorrect: This keeps AppSync waiting.
Even with increased timeout, synchronous invocations would block AppSync responses.
Option C: SQS queue + polling Lambda
Possible but too complex for this use case.
Requires additional infrastructure: queue + mapping + custom logic.
Higher operational overhead compared to built-in AppSync Event Mode.
Option D: Enable caching in AppSync
Irrelevant: AppSync cache is for optimizing repeated read queries, not for async workflows.
Asynchronous Lambda with AppSync:
https://docs.aws.amazon.com/appsync/latest/devguide/resolver- mapping-template-reference-lambda.html#async-lambda-invocation Lambda as AppSync Data Source:
https://docs.aws.amazon.com/appsync/latest/devguide/tutorial-lambda- resolvers.html
Event Mode Docs: https://docs.aws.amazon.com/appsync/latest/devguide/lambda-resolvers.html#event- invocation-mode

NEW QUESTION: 94

A developer needs to migrate an online retail application to AWS to handle an anticipated increase in traffic.
The application currently runs on two servers: one server for the web application and another server for the database. The web server renders webpages and manages session state in memory. The database server hosts a MySQL database that contains order details. When traffic to the application is heavy, the memory usage for the web server approaches 100% and the application slows down considerably.
The developer has found that most of the memory increase and performance decrease is related to the load of managing additional user sessions. For the web server migration, the developer will use Amazon EC2 instances with an Auto Scaling group behind an Application Load Balancer.
Which additional set of changes should the developer make to the application to improve the application's performance?
A. Use an EC2 instance to host the MySQL database. Store the session data and the application data in the MySQL database.
B. Use Amazon ElastiCache for Memcached to store and manage the session data. Use an Amazon RDS for MySQL DB instance to store the application data.
C. Use Amazon ElastiCache for Memcached to store and manage the session data and the application data.
D. Use the EC2 instance store to manage the session data. Use an Amazon RDS for MySQL DB instance to store the application data.
Answer: B (LEAVE A REPLY)
Using Amazon ElastiCache for Memcached to store and manage the session data will reduce the memory load and improve the performance of the web server. Using Amazon RDS for MySQL DB instance to store the application data will provide a scalable, reliable, and managed database service. Option A is not optimal because it does not address the memory issue of the web server. Option C is not optimal because it does not provide a persistent storage for the application data. Option D is not optimal because it does not provide a high availability and durability for the session data.
References: Amazon ElastiCache, Amazon RDS

NEW QUESTION: 95
A developer is working on an application that uses a microservices architecture. The developer needs to grant an AWS Lambda function access to make calls to an Amazon API Gateway HTTP API. The HTTP API requires authentication.
Which solution will meet these requirements?
A. Create a JSON Web Token (JWT) authorizer for the HTTP API. Create a new IAM identity provider (IdP). Configure the Lambda function's execution role to allow the Lambda function to access the IdP.
B. Enable IAM authorization for the HTTP API route. Configure the Lambda function's IAM role policy to allow the Lambda function to access the API.

C. Configure the HTTP API route to require API keys. Add an API key to the Lambda function environment variables.
D. Create a resource-based policy for the Lambda function that allows the Lambda function to access the HTTP API.
Answer: B (LEAVE A REPLY)
For service-to-service calls inside AWS, the simplest and most secure way to authenticate a caller (Lambda) to an API Gateway HTTP API is to use IAM authorization. When IAM auth is enabled on a route, API Gateway requires requests to be signed with AWS Signature Version 4 (SigV4). The calling Lambda function can sign the request using the AWS SDK (or SigV4 utilities) and its execution role credentials.
With this approach, you grant the Lambda function's execution role explicit permission to invoke the API by allowing the execute-api:Invoke action on the API's ARN. API Gateway then evaluates the SigV4-signed request against IAM to decide whether to authorize the call. This creates a clean, auditable permission model and avoids embedding static secrets.
Option A is incorrect and overly complex: JWT authorizers are typically used for end-user or workload identities from an OIDC/JWT provider; creating an IAM IdP is not how you enable JWT authorizers for HTTP APIs, and it does not directly solve Lambda-to-API authentication. Option C (API keys) is not considered an authentication mechanism for HTTP APIs in the same way; API keys are primarily for usage plans/throttling (and are more common with REST APIs). Storing an API key in environment variables is also weaker than IAM-based, short-lived credentials. Option D is backwards: a Lambda resource-based policy controls who can invoke the Lambda function, not who can call an API Gateway endpoint.
Therefore, B is the correct solution: enable IAM authorization on the HTTP API route and grant the Lambda function's IAM role permission to invoke the API (and sign requests with SigV4).

NEW QUESTION: 96
A company developed an API application on AWS by using Amazon CloudFront, Amazon API Gateway, and AWS Lambda. The API has a minimum of four requests every second.
A developer notices that many API users run the same query by using the POST method.
The developer wants to cache the POST request to optimize the API resources.
Which solution will meet these requirements?
A. Configure the CloudFront cache. Update the application to return cached content based upon the default request headers.
B. Override the cache method in the selected stage of API Gateway. Select the POST method.
C. Save the latest request response in Lambda /tmp directory. Update the Lambda function to check the
/tmp directory.

D. Save the latest request in AWS Systems Manager Parameter Store. Modify the Lambda function to take the latest request response from Parameter Store.

Answer: (SHOW ANSWER)

Amazon API Gateway provides tools for creating and documenting web APIs that route HTTP requests to Lambda functions2. You can secure access to your API with authentication and authorization controls. Your APIs can serve traffic over the internet or can be accessible only within your VPC2. You can override the cache method in the selected stage of API Gateway2. Therefore, option B is correct.

NEW QUESTION: 97

A developer has an application that is composed of many different AWS Lambda functions. The Lambda functions all use some of the same dependencies. To avoid security issues the developer is constantly updating the dependencies of all of the Lambda functions. The result is duplicated effort to reach function.

How can the developer keep the dependencies of the Lambda functions up to date with the LEAST additional complexity?

A. Define a maintenance window for the Lambda functions to ensure that the functions get updated copies of the dependencies.

B. Upgrade the Lambda functions to the most recent runtime version.

C. Define a Lambda layer that contains all of the shared dependencies.

D. Use an AWS CodeCommit repository to host the dependencies in a centralized location.

Answer: (SHOW ANSWER)

This solution allows the developer to keep the dependencies of the Lambda functions up to date with the least additional complexity because it eliminates the need to update each function individually. A Lambda layer is a ZIP archive that contains libraries, custom runtimes, or other dependencies. The developer can create a layer that contains all of the shared dependencies and attach it to multiple Lambda functions. When the developer updates the layer, all of the functions that use the layer will have access to the latest version of the dependencies.

Reference: [AWS Lambda layers]

NEW QUESTION: 98

A developer is building an application that stores sensitive data files in an Amazon S3 bucket. Company security policies require that files be encrypted by using AWS KMS keys. An application in a second AWS account must access the files.

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

A. Encrypt the files using server-side encryption with AWS KMS (SSE-KMS) and an AWS-managed KMS key.

B. Create an S3 bucket policy that allows access from the second AWS account.

C. Update the AWS KMS key policy to allow access from the second AWS account.

D. Create an IAM role that trusts the Amazon S3 service principal.

E. Encrypt the files using server-side encryption with AWS KMS and a customer-managed KMS key.
F. Configure default bucket encryption with SSE-S3.
Answer: B,C,E (LEAVE A REPLY)
When sharing SSE-KMS-encrypted S3 objects across AWS accounts, AWS documentation states that three elements are required:
* A customer-managed KMS key (Option E), because AWS-managed keys cannot be shared across accounts.
* An S3 bucket policy (Option B) that grants the second account permission to access the objects.
* A KMS key policy (Option C) that explicitly allows the second account to use the KMS key for decrypt operations.
Option A is incorrect because AWS-managed KMS keys cannot be shared. Option F uses SSE-S3, which does not use KMS and violates the encryption requirement. Option D is irrelevant because cross-account access does not require an S3 service role.
Therefore, the correct combination is B, C, and E.

NEW QUESTION: 99
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: (SHOW ANSWER)
Comprehensive Detailed Step by Step Explanation with All AWS Developer References:
The AWS Serverless Application Model (SAM) includes features for local testing and debugging of AWS Lambda functions. One of the most efficient ways to generate test payloads that match actual AWS event structures is by using the sam local generate-event command.
* sam local generate-event: This command allows developers to create pre-configured test event payloads for various AWS services (e.g., S3, API Gateway, SNS). These generated events accurately reflect the format that the service would use in a live environment, reducing the manual work required to create these events from scratch.
* Operational Overhead: This approach reduces overhead since the developer does not need to manually create or maintain test events. It ensures that the structure is correct and up-to-date with the latest AWS standards.

* Alternatives:
* Option A suggests using shareable test events, but manually creating or sharing these events introduces more overhead.
* Option B and C both involve manually storing and maintaining test events, which adds unnecessary complexity compared to using sam local generate-event.
AWS SAM CLI documentation

NEW QUESTION: 100
A developer needs to deploy an application running on AWS Fargate using Amazon ECS The application has environment variables that must be passed to a container for the application to initialize.
How should the environment variables be passed to the container?
A. Define an array that includes the environment variables under the environment parameter within the service definition.
B. Define an array that includes the environment variables under the environment parameter within the task definition.
C. Define an array that includes the environment variables under the entryPoint parameter within the task definition.
D. Define an array that includes the environment variables under the entryPoint parameter within the service definition.
Answer: B (LEAVE A REPLY)
This solution allows the environment variables to be passed to the container when it is launched by AWS Fargate using Amazon ECS. The task definition is a text file that describes one or more containers that form an application. It contains various parameters for configuring the containers, such as CPU and memory requirements, network mode, and environment variables. The environment parameter is an array of key-value pairs that specify environment variables to pass to a container. Defining an array that includes the environment variables under the entryPoint parameter within the task definition will not pass them to the container, but use them as command-line arguments for overriding the default entry point of a container.
Defining an array that includes the environment variables under the environment or entryPoint parameter within the service definition will not pass them to the container, but cause an error because these parameters are not valid for a service definition.
Reference: [Task Definition Parameters], [Environment Variables]

NEW QUESTION: 101
A developer needs to build an AWS CloudFormation template that self-populates the AWS Region variable that deploys the CloudFormation template What is the MOST operationally efficient way to determine the Region in which the template is being deployed?
A. Use the AWS:.Region pseudo parameter
B. Require the Region as a CloudFormation parameter

C. Find the Region from the AWS::StackId pseudo parameter by using the Fn::Split intrinsic function
D. Dynamically import the Region by referencing the relevant parameter in AWS Systems Manager Parameter Store
Answer: (SHOW ANSWER)
Pseudo Parameters: CloudFormation provides pseudo parameters that reference runtime context, including the current AWS Region.
Operational Efficiency: The AWS::Region pseudo parameter offers the most direct and self-contained way to obtain the Region dynamically within the template.
References:
CloudFormation Pseudo Parameters:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide
/pseudo-parameter-reference.html

NEW QUESTION: 102
A developer is designing a fault-tolerant environment where client sessions will be saved. How can the developer ensure that no sessions are lost if an Amazon EC2 instance fails?
A. Use Amazon DynamoDB to perform scalable session handling.
B. Use Elastic Load Balancer connection draining to stop sending requests to failing instances.
C. Use sticky sessions with an Elastic Load Balancer target group.
D. Use Amazon SOS to save session data.
Answer: (SHOW ANSWER)

NEW QUESTION: 103
A company has an application that is hosted on Amazon EC2 instances The application stores objects in an Amazon S3 bucket and allows users to download objects from the S3 bucket A developer turns on S3 Block Public Access for the S3 bucket After this change, users report errors when they attempt to download objects The developer needs to implement a solution so that only users who are signed in to the application can access objects in the S3 bucket.
Which combination of steps will meet these requirements in the MOST secure way? (Select TWO.)
A. Create an EC2 instance profile and role with an appropriate policy Associate the role with the EC2 instances
B. Create an 1AM user with an appropriate policy. Store the access key ID and secret access key on the EC2 instances
C. Modify the application to use the S3 GeneratePresignedUrl API call
D. Modify the application to use the S3 GetObject API call and to return the object handle to the user
E. Modify the application to delegate requests to the S3 bucket.

Answer: A,C (LEAVE A REPLY)

* IAM Roles for EC2 (A): The most secure way to provide AWS permissions from EC2.
* Create a role with a policy allowing s3:GetObject on the specific bucket.
* Attach the role to an instance profile and associate that profile with your instances.
* Pre-signed URLs (C): Temporary, authenticated URLs for specific S3 actions.
* Modify the app to use the AWS SDK to call GeneratePresignedUrl.
* Embed these URLs when a user is properly logged in, allowing download access.

References:

IAM Roles for EC2:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html
Generating Presigned URLs: https://docs.aws.amazon.com/AmazonS3/latest/userguide/ShareObjectPreSignedURL.htm

NEW QUESTION: 104

A developer is building a process flow that invokes two AWS Lambda functions. The Lambda functions write logs to Amazon CloudWatch. Each run of the process has a unique request ID that flows to both Lambda functions.

The developer encounters a failure in the process flow. The developer wants to use the request IDs to analyze the flow logs.

Which solution will meet these requirements with the LEAST development effort?

A. Use an AWS SDK to query the logs from Amazon CloudWatch.

B. Export the log data for a specific time range to an Amazon S3 bucket. Use Amazon Athena to query the S3 bucket.

C. Use Amazon CloudWatch Logs Insights to query log groups for the Lambda functions. Filter on the request IDs.

D. Use Amazon CloudWatch Live Tail to examine log groups for both Lambda functions, and check for the error.

Answer: C (LEAVE A REPLY)

The requirement is to analyze existing Lambda logs using a request ID that appears in both functions, and to do so with the least development effort. Amazon CloudWatch Logs Insights is purpose-built for interactive log analysis: it lets you run ad-hoc queries over CloudWatch log groups using a query language that supports filtering, parsing, sorting, and aggregation. This means you can quickly search across the relevant Lambda log groups and filter results to only entries that contain a specific request ID.

With Logs Insights, the developer can select both Lambda log groups, set a time range around the failure, and run a query such as filtering on the request ID field or matching the request ID string in the message. This provides a fast way to reconstruct the end-to-end flow and identify where the failure occurred-without writing any custom code, building data pipelines, or exporting logs.

Option A (using an AWS SDK) requires writing and maintaining code to call CloudWatch Logs APIs, handle pagination, time windows, filtering, and correlation across groups-more

effort than necessary. Option B (export to S3 + Athena) adds significant setup (export tasks, S3 storage, Athena tables/partitions) and is better suited to large-scale analytics or long-term archival, not quick troubleshooting. Option D (Live Tail) is primarily for real-time streaming of new log events; it is less suitable for retrospective investigation of an incident that already happened, and it doesn't provide the same structured querying and cross-log-group correlation capabilities as Logs Insights.
Therefore, CloudWatch Logs Insights (C) is the most direct, AWS-native, low-effort solution to query both Lambda log groups and filter by request IDs to troubleshoot the failing process flow.

NEW QUESTION: 105
A developer has written the following IAM policy to provide access to an Amazon S3 bucket:

Which access does the policy allow regarding the s3:GetObject and s3:PutObject actions?
A. Access on all buckets except the "DOC-EXAMPLE-BUCKET" bucket
B. Access on all buckets that start with "DOC-EXAMPLE-BUCKET" except the "DOC-EXAMPLE- BUCKET/secrets" bucket
C. Access on all objects in the "DOC-EXAMPLE-BUCKET" bucket along with access to all S3 actions for objects in the "DOC-EXAMPLE-BUCKET" bucket that start with "secrets"
D. Access on all objects in the "DOC-EXAMPLE-BUCKET" bucket except on objects that start with
"secrets"
Answer: D (LEAVE A REPLY)
The IAM policy shown in the image is a resource-based policy that grants or denies access to an S3 bucket based on certain conditions. The first statement allows access to any S3 action on any object in the "DOC- EXAMPLE-BUCKET" bucket when the request is made over HTTPS (the value of aws:SecureTransport is true). The second statement denies access to the s3:GetObject and s3:PutObject actions on any object in the

"DOC-EXAMPLE-BUCKET/secrets" prefix when the request is made over HTTP (the value of aws:
SecureTransport is false). Therefore, the policy allows access on all objects in the "DOC-EXAMPLE- BUCKET" bucket except on objects that start with "secrets". Reference: Using IAM policies for Amazon S3

NEW QUESTION: 106
A company hosts a client-side web application for one of its subsidiaries on Amazon S3. The web application can be accessed through Amazon CloudFront from https://www.example.com. After a successful rollout, the company wants to host three more client-side web applications for its remaining subsidiaries on three separate S3 buckets.
To achieve this goal, a developer moves all the common JavaScript files and web fonts to a central S3 bucket that serves the web applications. However, during testing, the developer notices that the browser blocks the JavaScript files and web fonts.
What should the developer do to prevent the browser from blocking the JavaScript files and web fonts?
A. Create four access points that allow access to the central S3 bucket. Assign an access point to each web application bucket.
B. Create a bucket policy that allows access to the central S3 bucket. Attach the bucket policy to the central S3 bucket.
C. Create a cross-origin resource sharing (CORS) configuration that allows access to the central S3 bucket.
Add the CORS configuration to the central S3 bucket.
D. Create a Content-MD5 header that provides a message integrity check for the central S3 bucket. Insert the Content-MD5 header for each web application request.
Answer: C (LEAVE A REPLY)
Browsers enforce the Same-Origin Policy, which restricts web pages from requesting resources (like JavaScript files, fonts, and other assets) from a different domain/origin unless the target explicitly allows it.
When the developer moved shared JavaScript and web fonts into a separate "central" S3 bucket, the applications likely began fetching these assets from a different origin than the app's primary origin (for example, different domain, subdomain, or CloudFront distribution). As a result, the browser blocks those requests unless the response includes the appropriate CORS headers.
AWS documentation for Amazon S3 explains that to allow cross-origin requests from browsers, you must configure Cross-Origin Resource Sharing (CORS) on the S3 bucket that serves the resources. A CORS configuration defines allowed origins, methods (such as GET), and headers. For fonts in particular, browsers are strict and commonly require CORS headers to load fonts across origins.

Option C directly addresses this by adding a CORS policy to the central bucket, allowing the subsidiary application origins to retrieve JS and font files without being blocked.
Option B (bucket policy) controls authorization at the AWS request level, but it does not instruct browsers to allow cross-origin access. Even if the request is authorized, the browser can still block it if CORS headers are missing.
Option A (access points) is also about access management and does not solve browser CORS enforcement.
Option D (Content-MD5) is for integrity checking and is unrelated to cross-origin browser blocking.

NEW QUESTION: 107
A company has an AWS Step Functions state machine named myStateMachine. The company configured a service role for Step Functions. The developer must ensure that only the myStateMachine state machine can assume the service role.
A. "Condition": { "ArnLike": { "aws":"arn:aws:states:ap-south-1:111111111111:stateMachine" } }
B. "Condition": { "ArnLike": { "aws":"arn:aws:states:ap-south-1:*:stateMachine" } }
Answer: A (LEAVE A REPLY)
Comprehensive Detailed Step by Step Explanation with All AWS Developer References:
To ensure that only a specific AWS Step Functions state machine (myStateMachine) can assume the service role, you must configure the correct trust policy in AWS IAM.
Trust Policies: Trust policies determine which entities (services or users) are allowed to assume the role. In this case, we want to restrict the trust policy to only allow the specific state machine (myStateMachine) to assume the role.
Using ArnLike: The condition "ArnLike" is used to specify that the SourceArn (which refers to the ARN of the entity assuming the role) must match a specific ARN. Option A specifies the exact ARN of the myStateMachine state machine, ensuring that only this state machine can assume the role.
Option B: This option is incorrect because it uses a wildcard (*) for the account ID, which would allow any state machine in the ap-south-1 region to assume the role, not just the specific one.
AWS Step Functions IAM Policies

NEW QUESTION: 108
A developer is troubleshooting an application in an integration environment. In the application, an Amazon Simple Queue Service (Amazon SQS) queue consumes messages and then an AWS Lambda function processes the messages. The Lambda function transforms the messages and makes an API call to a third-party service.
There has been an increase in application usage. The third-party API frequently returns an HTTP 429 Too Many Requests error message. The error message prevents a significant number of messages from being processed successfully.
How can the developer resolve this issue?
A. Increase the SQS event source's batch size setting.
B. Configure provisioned concurrency for the Lambda function based on the third-party API's documented rate limits.
C. Increase the retry attempts and maximum event age in the Lambda function's asynchronous configuration.
D. Configure maximum concurrency on the SQS event source based on the third-party service's documented rate limits.
Answer: D (LEAVE A REPLY)
Maximum concurrency for SQS as an event source allows customers to control the maximum concurrent invokes by the SQS event source1. When multiple SQS event sources are configured to a function, customers can control the maximum concurrent invokes of individual SQS event source1.
In this scenario, the developer needs to resolve the issue of the third-party API frequently returning an HTTP
429 Too Many Requests error message, which prevents a significant number of messages from being processed successfully. To achieve this, the developer can follow these steps:
Find out the documented rate limits of the third-party API, which specify how many requests can be made in a given time period.
Configure maximum concurrency on the SQS event source based on the rate limits of the third-party API.
This will limit the number of concurrent invokes by the SQS event source and prevent exceeding the rate limits of the third-party API.
Test and monitor the application performance and adjust the maximum concurrency value as needed.
By using this solution, the developer can reduce the frequency of HTTP 429 errors and improve the message processing success rate. The developer can also avoid throttling or blocking by the third-party API.

NEW QUESTION: 109

A team is developing an application that is deployed on Amazon EC2 instances. During testing, the team receives an error. The EC2 instances are unable to access an Amazon S3 bucket.
Which steps should the team take to troubleshoot this issue? (Select TWO.)
A. Check whether the policy that is assigned to the JAM role that is attached to the EC2 instances grants access to Amazon S3.
B. Check the S3 Lifecycle policy to validate the permissions that are assigned to the S3 bucket.
C. Check the security groups that are assigned to the EC2 instances. Make sure that a rule is not blocking the access to Amazon S3.
D. Check whether the policy that is assigned to the 1AM user that is attached to the EC2 instances grants access to Amazon S3.
E. Check the S3 bucket policy to validate the access permissions for the S3 bucket.
Answer: A,E (LEAVE A REPLY)

NEW QUESTION: 110
A company is running Amazon EC2 instances in multiple AWS accounts. A developer needs to implement an application that collects all the lifecycle events of the EC2 instances. The application needs to store the lifecycle events in a single Amazon Simple Queue Service (Amazon SQS) queue in the company's main AWS account for further processing.
Which solution will meet these requirements?
A. Configure Amazon EC2 to deliver the EC2 instance lifecycle events from all accounts to the Amazon EventBridge event bus of the main account. Add an EventBridge rule to the event bus of the main account that matches all EC2 instance lifecycle events. Add the SQS queue as a target of the rule.
B. Use the resource policies of the SQS queue in the main account to give each account permissions to write to that SQS queue. Add to the Amazon EventBridge event bus of each account an EventBridge rule that matches all EC2 instance lifecycle events. Add the SQS queue in the main account as a target of the rule.
C. Write an AWS Lambda function that scans through all EC2 instances in the company accounts to detect EC2 instance lifecycle changes. Configure the Lambda function to write a notification message to the SQS queue in the main account if the function detects an EC2 instance lifecycle change. Add an Amazon EventBridge scheduled rule that invokes the Lambda function every minute.
D. Configure the permissions on the main account event bus to receive events from all accounts. Create an Amazon EventBridge rule in each account to send all the EC2 instance lifecycle events to the main account event bus. Add an EventBridge rule to the main account event bus that matches all EC2 instance lifecycle events. Set the SQS queue as a target for the rule.
Answer: D (LEAVE A REPLY)

Amazon EC2 instances can send the state-change notification events to Amazon EventBridge. https://docs.
aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-instance-state-changes.html
Amazon EventBridge can send and receive events between event buses in AWS accounts. https://docs.aws.amazon.com/eventbridge
/latest/userguide/eb-cross-account.html

NEW QUESTION: 111
A developer updates an AWS Lambda function that is integrated with an Amazon API Gateway API. The API serves as the backend for a web application. The developer must test the updated Lambda function without affecting production users.
Which solution will meet these requirements in the MOST operationally efficient way?
A. Create a canary deployment on the existing API stage and test by using the production URL.
B. Change the API endpoint type to private and test by using the production URL.
C. Create a new API Gateway stage for testing and use stage variables to route traffic to the updated Lambda function.
D. Deploy a separate CloudFormation stack that duplicates the production API and Lambda function.
Answer: C (LEAVE A REPLY)
Amazon API Gateway supports multiple stages, allowing developers to deploy and test different versions of an API independently. By creating a new test stage, the developer can route traffic to an updated Lambda function without impacting production users.
AWS documentation recommends using stage variables to dynamically reference different Lambda function versions or aliases. This approach enables safe testing while reusing the same API configuration.
Canary deployments (Option A) expose a portion of production users to the new code, which violates the requirement. Changing the endpoint type (Option B) disrupts production. Duplicating the entire stack (Option D) introduces unnecessary complexity and overhead. Using a separate test stage is the most efficient, low-risk, and AWS-recommended solution.

NEW QUESTION: 112
A company caches session information for a web application in an Amazon DynamoDB table. The company wants an automated way to delete old items from the table.
What is the simplest way to do this?
A. Add an attribute with the expiration time; enable the Time To Live feature based on that attribute.
B. Write a script that deletes old records; schedule the script as a cron job on an Amazon EC2 instance.
C. Add an attribute with the expiration time; name the attribute ItemExpiration.

D. Each day, create a new table to hold session data; delete the previous day's table.
Answer: (SHOW ANSWER)

NEW QUESTION: 113
A company is building a serverless application on AWS. The application uses Amazon API Gateway and AWS Lambda. The company wants to deploy the application to its development, test, and production environments.
Which solution will meet these requirements with the LEAST development effort?
A. Use API Gateway stage variables and create Lambda aliases to reference environment-specific resources.
B. Use Amazon ECS to deploy the application to the environments.
C. Duplicate the code for each environment. Deploy the code to a separate API Gateway stage.
D. Use AWS Elastic Beanstalk to deploy the application to the environments.
Answer: A (LEAVE A REPLY)
Requirement Summary:
Deploy serverless application using:
API Gateway
AWS Lambda
Need dev, test, and prod environments
Want least development effort
Evaluate Options:
Option A: API Gateway stage variables + Lambda aliases
Most efficient and scalable
API Gateway supports stage variables (like env)
Lambda supports aliases (e.g., dev, test, prod)
You can configure each stage to point to a different alias of the same function version
Enables versioning, isolation, and low effort management Option B: Use Amazon ECS
Overkill for a serverless setup ECS is container-based, not serverless Introduces unnecessary complexity Option C: Duplicate code for each environment High operational overhead and poor maintainability Option D: Use Elastic Beanstalk Not applicable: Elastic Beanstalk is for traditional app hosting, not optimal for Lambda + API Gateway Lambda Aliases: https://docs.aws.amazon.com/lambda/latest/dg/configuration-aliases.html API Gateway Stage Variables:
https://docs.aws.amazon.com/apigateway/latest/developerguide/stage-variables.
html

NEW QUESTION: 114
An application development team decides to use AWS X-Ray to monitor application code to analyze performance and perform root cause analysis.
What does the team need to do to begin using X-Ray? (Select TWO.)

A. Log instrumentation output into an Amazon SQS queue.
B. Use a visualization tool to view application traces.
C. Instrument application code using the AWS SDK.
D. Install the X-Ray agent on the application servers.
E. Create an Amazon DynamoDB table to store the trace logs.
Answer: C,D (LEAVE A REPLY)
To begin using AWS X-Ray, the team must (1) instrument the application to emit trace segments
/subsegments and (2) ensure there is a component that can send trace data to the X-Ray service.
Instrumenting code is typically done with the AWS X-Ray SDK (language-specific). This adds tracing to inbound requests and downstream calls, creates segments/subsegments, and attaches annotations/metadata.
That corresponds to Option C.
For many compute environments (especially EC2, ECS on EC2, and on-prem servers), the application sends trace data via the X-Ray daemon/agent running locally. The daemon batches segments and forwards them to the X-Ray service endpoint. That corresponds to Option D.
Option B is helpful after traces exist (using the X-Ray console or CloudWatch ServiceLens), but it is not a prerequisite to "begin using" X-Ray.
Option A is unrelated. X-Ray does not require SQS for instrumentation output.
Option E is incorrect because X-Ray stores trace data in its managed backend; you do not create a DynamoDB table for trace logs.
Therefore, the team needs to instrument the code with the X-Ray SDK and install/run the X-Ray agent
/daemon on the servers where the application runs.

NEW QUESTION: 115
A company is working on a new serverless application. A developer needs to find an automated way to deploy AWS Lambda functions and the dependent Infrastructure with minimum coding effort. The application also needs to be reliable.
Which method will meet these requirements with the LEAST operational overhead?
A. Build the application by using the AWS Serverless Application Model (AWS SAM). Use a continuous integration and continuous delivery (CI/CD) pipeline and the SAM CLI to deploy the Lambda functions.
B. Build a container for each Lambda function. Store the container images in AWS CodeArtifact. Deploy the containers as Lambda functions by using the AWS CLI in a continuous integration and continuous delivery (CI/CD) pipeline.
C. Build the application by using shell scripts to create .zip files for each Lambda function. Manually upload the .zip files to the AWS Management Console.

D. Build the application by using shell scripts to create .zip files for each Lambda function. Upload the .zip files. Deploy the .zip files as Lambda functions by using the AWS CLI in a continuous integration and continuous delivery (CI/CD) pipeline.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 116
A developer is working on an application that will store protected health information (PHI) in an Amazon RDS database. The developer applies encryption to the database. The developer must also encrypt the PHI data separately to prevent administrators from accessing the data. Because some of the PHI data files are large, the developer must encrypt the PHI data in the application locally before saving the data to the database.
Which solution will meet these requirements in the MOST secure way?
A. Create an AWS KMS customer managed key. Use the KMS Encrypt operation to encrypt the PHI data before storing the PHI data in the database.
B. Generate a 256-bit AES encryption key. Store the key in base64-encoded format in the application source code. Use the encryption key to encrypt the PHI data before storing the PHI data in the database.
C. Configure the database to use an AWS KMS managed key for encryption.
D. Create an AWS KMS customer managed key. Use envelope encryption to encrypt the PHI data. Store the encrypted key in the same database record that stores the PHI data.
Answer: D (LEAVE A REPLY)
The requirement is application-level encryption (client-side) in addition to RDS encryption at rest, specifically to prevent database administrators from viewing PHI. Also, some PHI objects are large, so the solution must encrypt locally without size limitations.
AWS best practice for large data encryption with KMS is envelope encryption. With envelope encryption, the application generates a data encryption key (DEK) (typically obtained from KMS using GenerateDataKey).
The application uses the plaintext DEK locally with a symmetric algorithm (such as AES-GCM) to encrypt the large PHI payload efficiently. The DEK itself is then protected by encrypting it with a KMS customer managed key (CMK), producing an encrypted data key that can be safely stored alongside the ciphertext.
Only principals with permission to use the CMK can decrypt the DEK and then decrypt the PHI data. This ensures that even database administrators who can access the RDS data cannot read PHI without KMS authorization.
Option A is not suitable because the KMS Encrypt operation is intended for small payloads (KMS has size limits for direct encryption), making it inappropriate for large PHI files.
Option B is insecure because embedding encryption keys in source code violates key management best practices and makes rotation and compromise containment difficult.
Option C provides only storage-level encryption for the database volume; it does not ensure that PHI remains unreadable to administrators who can access the database content.

Therefore, using envelope encryption with a KMS customer managed key and storing the encrypted data key with the record is the most secure solution.

NEW QUESTION: 117
A developer is working on an AWS Lambda function that accesses Amazon DynamoDB. The Lambda function must retrieve an item and update some of its attributes, or create the item if it does not exist. The Lambda function has access to the primary key.
Which IAM permissions should the developer request for the Lambda function to achieve this functionality?
A)
B)
C)
D)
A. Option A
B. Option B
C. Option C
D. Option D
Answer: B (LEAVE A REPLY)
Requirement Summary:
Lambda function:
Retrieves an item from DynamoDB
Updates its attributes or creates it if it does not exist
Has primary key
Needs minimum required IAM permissions
Analysis of Required Permissions:
To get an item and update it or create it if it doesn't exist, the Lambda function must use the **PutItem** or
UpdateItem API, and read with **GetItem**.
Valid API options:
GetItem: Read the item from the table.
UpdateItem: Update existing item attributes or insert if it doesn't exist (with UpdateExpression and ConditionExpression).
When UpdateItem is used without a conditional check for existence, it can create a new item if it does not exist (acts like upsert).
DescribeTable: (Optional) Used if you need table metadata (not strictly required here).
Evaluate the Choices:
Option A:
DeleteItem - Not needed
GetItem -
PutItem - (can create/replace but not partial update)

Close, but PutItem overwrites the full item, not update-in-place. Acceptable, but UpdateItem is better suited.
Option B:
UpdateItem - Required to modify attributes or insert new item
GetItem - Required to check existence or read data
DescribeTable - Optional, but not harmful
BEST FIT - Matches the update-or-create logic.
Option C:
GetRecords - This is used for DynamoDB Streams, not standard GetItem
PutItem -
UpdateTable - Used to change table settings, not data manipulation
Incorrect usage context
Option D:
UpdateItem, GetItem, PutItem - all valid
But UpdateItem alone is sufficient; including PutItem might not be necessary Also, image is faded (possibly invalid), and it's redundant UpdateItem API:
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_UpdateItem.html
PutItem vs UpdateItem:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide
/WorkingWithItems.html
IAM actions for DynamoDB: https://docs.aws.amazon.com/service-authorization/latest/reference
/list_amazondynamodb.html

NEW QUESTION: 118
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,E (LEAVE A REPLY)
Amazon EventBridge: A service that reacts to events from various AWS sources, including S3. Rules define which events trigger actions (like invoking Lambda functions).

S3 Object Created Events: EventBridge can detect these, providing seamless integration for automated CSV processing.
S3 Lifecycle Rules: Allow for actions based on object age or prefixes. These can directly trigger Lambda functions for file processing.
References:
Amazon EventBridge Documentation: https://docs.aws.amazon.com/eventbridge/ Working with S3 Event Notifications: https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventNotifications.html
S3 Lifecycle Configuration: https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html

NEW QUESTION: 119
A developer is building an application that will use an Amazon API Gateway API with an AWS Lambda backend. The team that will develop the frontend requires immediate access to the API endpoints to build the UI. To prepare the backend application for integration, the developer needs to set up endpoints. The endpoints need to return predefined HTTP status codes and JSON responses for the frontend team. The developer creates a method for an API resource.
Which solution will meet these requirements?
A. Set the integration type to AWS_PROXY. Provision Lambda functions to return hardcoded JSON data.
B. Set the integration type to MOCK. Configure the method's integration request and integration response to associate JSON responses with specific HTTP status codes.
C. Set the integration type to HTTP PROXY. Configure API Gateway to pass all requests to an external placeholder API, which the team will build.
D. Set the integration type to MOCK. Use a method request to define HTTP status codes. Use an integration request to define JSON responses.
Answer: (SHOW ANSWER)
To give the frontend team immediate access to stable endpoints without waiting for a real backend implementation, the simplest approach is to use API Gateway MOCK integration. A MOCK integration lets API Gateway generate a response directly, without invoking Lambda or any other backend. This is ideal for early UI development because you can return consistent, predefined payloads and status codes for each method.
With MOCK integration, you configure an integration request (often using a static mapping template) to produce a dummy request payload that triggers a response, and then you configure integration responses to return specific HTTP status codes (such as 200, 400, 500) along with predefined JSON bodies. You can also set response headers (for example, Content-Type: application/json) and use mapping templates to shape the returned JSON. This satisfies the requirement to "return predefined HTTP status codes

and JSON responses" while minimizing development effort and avoiding the need to deploy placeholder Lambda functions.
Option A (AWS_PROXY with Lambda) can work, but it requires creating and deploying Lambda functions and IAM permissions just to return static data-more effort than necessary. Option C (HTTP PROXY) depends on an external placeholder service, which adds another system to build and maintain. Option D is incorrect because HTTP status codes are not defined in the method request; they are defined through method responses and realized through integration responses in API Gateway's integration configuration. Therefore, B is the best solution: MOCK integration with integration request/response mappings provides quick, code-free stubs that unblock frontend development.

NEW QUESTION: 120
A company wants to automate part of its deployment process. A developer needs to automate the process of checking for and deleting unused resources that supported previously deployed stacks but that are no longer used.
The company has a central application that uses the AWS Cloud Development Kit (AWS CDK) to manage all deployment stacks. The stacks are spread out across multiple accounts. The developer's solution must integrate as seamlessly as possible within the current deployment process.
Which solution will meet these requirements with the LEAST amount of configuration?
A. In the central AWS CDK application, write a handler function in the code that uses AWS SDK calls to check for and delete unused resources. Create an AWS CloudPormation template from a JSON file. Use the template to attach the function code to an AWS Lambda function and lo invoke the Lambda function when the deployment slack runs.
B. In the central AWS CDK application. write a handler function in the code that uses AWS SDK calls to check for and delete unused resources. Create an AWS CDK custom resource Use the custom resource to attach the function code to an AWS Lambda function and to invoke the Lambda function when the deployment stack runs.
C. In the central AWS CDK, write a handler function m the code that uses AWS SDK calls to check for and delete unused resources. Create an API in AWS Amplify Use the API to attach the function code to an AWS Lambda function and to invoke the Lambda function when the deployment stack runs.
D. In the AWS Lambda console write a handler function in the code that uses AWS SDK calls to check for and delete unused resources. Create an AWS CDK custom resource. Use the custom resource to import the Lambda function into the stack and to Invoke the Lambda function when the deployment stack runs.
Answer: B (LEAVE A REPLY)
This solution meets the requirements with the least amount of configuration because it uses a feature of AWS CDK that allows custom logic to be executed during stack deployment or deletion. The AWS Cloud Development Kit (AWS CDK) is a software development framework that allows you to define cloud infrastructure as code and

provision it through CloudFormation. An AWS CDK custom resource is a construct that enables you to create resources that are not natively supported by CloudFormation or perform tasks that are not supported by CloudFormation during stack deployment or deletion. The developer can write a handler function in the code that uses AWS SDK calls to check for and delete unused resources, and create an AWS CDK custom resource that attaches the function code to a Lambda function and invokes it when the deployment stack runs. This way, the developer can automate the cleanup process without requiring additional configuration or integration. Creating a CloudFormation template from a JSON file will require additional configuration and integration with the central AWS CDK application. Creating an API in AWS Amplify will require additional configuration and integration with the central AWS CDK application and may not provide optimal performance or availability. Writing a handler function in the AWS Lambda console will require additional configuration and integration with the central AWS CDK application.
Reference: [AWS Cloud Development Kit (CDK)], [Custom Resources]

NEW QUESTION: 121
A developer built an application that calls an external API to obtain data, processes the data, and saves the result to Amazon S3. The developer built a container image with all of the necessary dependencies to run the application as a container.
The application runs locally and requires minimal CPU and RAM resources. The developer has created an Amazon ECS cluster. The developer needs to run the application hourly in Amazon ECS.
Which solution will meet these requirements with the LEAST amount of infrastructure management overhead?
A. Add a capacity provider to manage instances.
B. Add an Amazon EC2 instance that runs the application.
C. Define a task definition with an AWS Fargate launch type.
D. Create an Amazon ECS cluster and add the managed node groups feature to run the application.
Answer: C (LEAVE A REPLY)
To run a containerized job hourly in ECS with minimal infrastructure management, the best choice is AWS Fargate. Fargate is a serverless compute engine for containers that runs ECS tasks without the developer needing to provision, patch, or scale EC2 instances.
With option C, the developer defines an ECS task definition using the Fargate launch type and schedules it (commonly via Amazon EventBridge Scheduler / EventBridge rule triggering RunTask). ECS handles the placement, networking, and execution of the task on managed infrastructure. This is the least operational overhead approach because there are no instances to manage and no capacity planning required, which fits an hourly job with small CPU/RAM needs.
Option A (capacity providers) is primarily for managing EC2 capacity for ECS and still requires underlying instances, which increases operational burden.

Option B runs the app on an EC2 instance, which requires instance lifecycle management, OS patching, scaling considerations, and failure handling.
Option D refers to managed node groups (an EKS concept) and does not apply cleanly to ECS. Even in a Kubernetes context, nodes still need management compared to Fargate.
Therefore, defining a Fargate task is the simplest and lowest-management solution.

NEW QUESTION: 122
An application stores user data in Amazon S3 buckets in multiple AWS Regions. A developer needs to implement a solution that analyzes the user data in the S3 buckets to find sensitive information. The analysis findings from all the S3 buckets must be available in the eu-west-2 Region.
Which solution will meet these requirements with the LEAST development effort?
A. Configure Amazon Inspector to generate findings. Use Amazon EventBridge to create rules that copy the findings to eu-west-2.
B. Configure Amazon Macie to generate findings and to publish the findings to AWS CloudTrail. Use a CloudTrail trail to copy the results to eu-west-2.
C. Configure Amazon Made to generate findings. Use Amazon EventBridge to create rules that copy the findings to eu-west-2.
D. Create an AWS Lambda function to generate findings. Program the Lambda function to send the findings to another S3 bucket in eu-west-2.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 123
A developer is preparing to deploy an AWS CloudFormation stack for an application from a template that includes an IAM user.
The developer needs to configure the application's resources to retain the IAM user after successful creation.
However, the developer also needs to configure the application to delete the IAM user if the stack rolls back.
A. Update CloudFormation template with the following deletion policy:AWSTemplateFormatVersion:
'2010-05-09' Resources: appUser: Type: AWS::IAM::User DeletionPolicy: Retain

B. Update CloudFormation template with the following deletion policy:AWSTemplateFormatVersion:
'2010-09-09' Resources: appUser: Type: AWS::IAM::User DeletionPolicy: RetainExceptOnCreate
C. Update the CloudFormation service role to include the following policy:{"Version": "2012-10-17"," Statement": [{"Effect": "Allow","Action": ["cloudformation:UpdateTerminationProtection"]," Resource": "*"}]}
D. Update the stack policy to include the following statements:{"Statement": [{"Effect": "Deny","Action":
"Update:*","Principal": "*","Resource": "*","Condition": {"StringEquals": {"ResourceType": "AWS::IAM::User"}}}]}
Answer: B (LEAVE A REPLY)
Why Option B is Correct:The RetainExceptOnCreate deletion policy ensures that the IAM user is retained after successful stack creation but is deleted if the stack creation fails or rolls back. This meets both requirements.
Why Other Options are Incorrect:
Option A: The Retain policy retains the resource regardless of stack status and does not delete the IAM user upon rollback.
Option C: Updating the service role for termination protection does not address the specific deletion behavior for the IAM user.
Option D: Stack policy controls updates, not resource deletion behavior during rollbacks.
AWS Documentation References:
CloudFormation DeletionPolicy Attribute

NEW QUESTION: 124
A developer must cache dependent artifacts from Maven Central, a public package repository, as part of an application's build pipeline. The build pipeline has an AWS CodeArtifact repository where artifacts of the build are published. The developer needs a solution that requires minimum changes to the build pipeline.
Which solution meets these requirements?
A. Create a new CodeArtifact repository that has an external connection to the public package repository.
B. Modify the existing CodeArtifact repository to associate an upstream repository with the public package repository.
C. Create a new CodeArtifact domain that contains a new repository that has an external connection to the public package repository.
D. Modify the CodeArtifact repository resource policy to allow artifacts to be fetched from the public package repository.
Answer: (SHOW ANSWER)

NEW QUESTION: 125

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 and trace all requests for requests that handle user interactions or transactions. Sample high-volume read-only requests at a higher rate.
B. Disable sampling and trace all requests for requests that handle user interactions or transactions. Sample high-volume read-only requests at a lower rate.
C. Disable sampling for high-volume read-only requests. Sample at a higher rate for all requests that handle user interactions or transactions.
D. Disable sampling for high-volume read-only requests. Sample at a lower rate for all requests that handle user interactions or transactions.
Answer: (SHOW ANSWER)

NEW QUESTION: 126
A developer is creating an application that includes an Amazon API Gateway REST API in the us-east-2 Region. The developer wants to use Amazon CloudFront and a custom domain name for the API. The developer has acquired an SSL/TLS certificate for the domain from a third-party provider.
How should the developer configure the custom domain for the application?
A. Import the SSL/TLS certificate into AWS Certificate Manager (ACM) in the same Region as the API.
Create a DNS A record for the custom domain.
B. Import the SSL/TLS certificate into CloudFront. Create a DNS CNAME record for the custom domain.
C. Import the SSL/TLS certificate into AWS Certificate Manager (ACM) in the same Region as the API.
Create a DNS CNAME record for the custom domain.
D. Import the SSL/TLS certificate into AWS Certificate Manager (ACM) in the us-east-1 Region. Create a DNS CNAME record for the custom domain.
Answer: (SHOW ANSWER)
Amazon API Gateway is a service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. Amazon CloudFront is a content delivery network

(CDN) service that can improve the performance and security of web applications. The developer can use CloudFront and a custom domain name for the API Gateway REST API. To do so, the developer needs to import the SSL/TLS certificate into AWS Certificate Manager (ACM) in the us-east-1 Region. This is because CloudFront requires certificates from ACM to be in this Region. The developer also needs to create a DNS CNAME record for the custom domain that points to the CloudFront distribution.
References:
* [What Is Amazon API Gateway? - Amazon API Gateway]
* [What Is Amazon CloudFront? - Amazon CloudFront]
* [Custom Domain Names for APIs - Amazon API Gateway]

NEW QUESTION: 127
A company is developing a web application that allows its employees to upload a profile picture to a private Amazon S3 bucket. There is no size limit for the profile pictures, which should be displayed every time an employee logs in. For security reasons, the pictures cannot be publicly accessible.
What is a viable long-term solution for this scenario?
A. Generate a presigned URL when a picture is uploaded. Save the URL in an Amazon DynamoDB table.
Return the URL to the browser when the employee logs in.
B. Save the picture's S3 key in an Amazon DynamoDB table. Create an Amazon S3 VPC endpoint to allow the employees to download pictures once they log in.
C. Encode a picture using base64. Save the base64 string in an Amazon DynamoDB table. Allow the browser to retrieve the string and convert it to a picture.
D. Save the picture's S3 key in an Amazon DynamoDB table. Use a function to generate a presigned URL every time an employee logs in. Return the URL to the browser.
Answer: D (LEAVE A REPLY)
The requirement is to keep profile pictures private in S3 while allowing the browser to display them each time an employee logs in. The standard AWS pattern is to store objects in a private bucket and provide time- limited access through S3 presigned URLs. A presigned URL grants temporary permission to perform a specific operation (such as GET) on a specific object, and it expires automatically after a configured duration.
This prevents public access while still allowing simple browser retrieval.
Option D is the viable long-term solution because it stores only the S3 object key (a stable identifier) in DynamoDB and generates a fresh presigned URL at login time. This is important because presigned URLs expire; storing a presigned URL (option A) would eventually store an expired URL and force extra complexity to refresh records. Generating a new presigned URL when needed ensures the link is always valid and keeps the expiration window tight, which improves security.
Option B is not appropriate for employee browsers in general: a VPC endpoint is for traffic from resources inside a VPC (EC2, Lambda in VPC, etc.). Employee browsers on the

public internet cannot use a private VPC endpoint to reach S3. Option C is not viable for "no size limit" images: DynamoDB has strict item size limits, base64 inflates data size, and storing large binary blobs in DynamoDB is inefficient and expensive compared to S3. Therefore, D best meets the requirement: keep images private in S3, store the S3 key in DynamoDB, and generate a presigned URL on each login to provide secure, temporary access for display.

NEW QUESTION: 128
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 (LEAVE A REPLY)
This solution will meet the requirements by using AWS Secrets Manager, which is a service that helps protect secrets such as database credentials by encrypting them with AWS Key Management Service (AWS KMS) and enabling automatic rotation of secrets. The developer can use an AWS Secrets Manager resource in AWS CloudFormation template, which enables creating and managing secrets as part of a CloudFormation stack. The developer can use an AWS::SecretsManager::Secret resource type to generate and rotate the password for accessing RDS database during deployment. The developer can also specify a RotationSchedule property for the secret resource, which defines how often to rotate the secret and which Lambda function to use for rotation logic. Option A is not optimal because it will use an AWS Lambda function as a CloudFormation custom resource, which may introduce additional complexity and overhead for creating and managing a custom resource and implementing rotation logic. Option B is not optimal because it will use an AWS Systems Manager Parameter Store resource with the SecureString data type, which does not support automatic rotation of secrets. Option C is not optimal because it will use a cron daemon on the application's host to generate and rotate the password, which may incur more costs and require more maintenance for running and securing a host.
References: [AWS Secrets Manager], [AWS::SecretsManager::Secret]

NEW QUESTION: 129

A social media application is experiencing high volumes of new user requests after a recent marketing campaign. The application is served by an Amazon RDS for MySQL instance. A solutions architect examines the database performance and notices high CPU usage and many "too many connections" errors that lead to failed requests on the database. The solutions architect needs to address the failed requests.
Which solution will meet this requirement?
A. Deploy an Amazon DynamoDB Accelerator (DAX) cluster. Configure the application to use the DAX cluster.
B. Deploy an RDS Proxy. Configure the application to use the RDS Proxy.
C. Migrate the database to an Amazon RDS for PostgreSQL instance.
D. Deploy an Amazon ElastiCache (Redis OSS) cluster. Configure the application to use the ElastiCache cluster.
Answer: (SHOW ANSWER)
* Why Option B is Correct:RDS Proxy manages database connections efficiently, reducing overhead on the RDS instance and mitigating "too many connections" errors.
* Why Other Options are Incorrect:
* Option A: DAX is for DynamoDB, not RDS.
* Option C: Migration to PostgreSQL does not address the current issue.
* Option D: ElastiCache is useful for caching but does not solve connection pool issues.
* AWS Documentation References:
* Amazon RDS Proxy

NEW QUESTION: 130
A developer has designed an application to store incoming data as JSON files in Amazon S3 objects. Custom business logic in an AWS Lambda function then transforms the objects, and the Lambda function loads the data into an Amazon DynamoDB table. Recently, the workload has experienced sudden and significant changes in traffic. The flow of data to the DynamoDB table is becoming throttled.
The developer needs to implement a solution to eliminate the throttling and load the data into the DynamoDB table more consistently.
Which solution will meet these requirements?
A. Refactor the Lambda function into two functions. Configure one function to store the data in the DynamoDB table. Configure the second function to process the data and update the items after the data is stored in DynamoDB. Create a DynamoDB stream to invoke the second function after the data is stored.
B. Turn on auto scaling for the DynamoDB table. Use Amazon CloudWatch to monitor the table's read and write capacity metrics and to track consumed capacity.
C. Refactor the Lambda function into two functions. Configure one function to transform the data and one function to load the data into the DynamoDB table. Create an Amazon Simple Queue Service (Amazon SQS) queue in between the functions to hold the items as messages and to invoke the second function.

D. Create an alias for the Lambda function. Configure provisioned concurrency for the application to use.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 131
A large company has its application components distributed across multiple AWS accounts. The company needs to collect and visualize trace data across these accounts. What should be used to meet these requirements?
A. AWS X-Ray
B. Amazon CloudWatch
C. Amazon VPC flow logs
D. Amazon OpenSearch Service
Answer: A (LEAVE A REPLY)

NEW QUESTION: 132
A company hosts its application on AWS. The application runs on an Amazon Elastic Container Service (Amazon ECS) cluster that uses AWS Fargate. The cluster runs behind an Application Load Balancer The application stores data in an Amazon Aurora database A developer encrypts and manages database credentials inside the application The company wants to use a more secure credential storage method and implement periodic credential rotation.
Which solution will meet these requirements with the LEAST operational overhead?
A. Migrate the secret credentials to Amazon RDS parameter groups. Encrypt the parameter by using an AWS Key Management Service (AWS KMS) key Turn on secret rotation. Use 1AM policies and roles to grant AWS KMS permissions to access Amazon RDS.
B. Migrate the credentials to AWS Systems Manager Parameter Store. Encrypt the parameter by using an AWS Key Management Service (AWS KMS) key. Turn on secret rotation. Use 1AM policies and roles to grant Amazon ECS Fargate permissions to access to AWS Secrets Manager
C. Migrate the credentials to ECS Fargate environment variables. Encrypt the credentials by using an AWS Key Management Service (AWS KMS) key Turn on secret rotation. Use 1AM policies and roles to grant Amazon ECS Fargate permissions to access to AWS Secrets Manager.
D. Migrate the credentials to AWS Secrets Manager. Encrypt the credentials by using an AWS Key Management Service (AWS KMS) key Turn on secret rotation Use 1AM policies and roles to grant Amazon ECS Fargate permissions to access to AWS Secrets Manager by using keys.
Answer: D (LEAVE A REPLY)
Secrets Management: AWS Secrets Manager is designed specifically for storing and managing sensitive credentials.

Built-in Rotation: Secrets Manager provides automatic secret rotation functionality, enhancing security posture significantly.
IAM Integration: IAM policies and roles grant fine-grained access to ECS Fargate, ensuring the principle of least privilege.
Reduced Overhead: This solution centralizes secrets management and automates rotation, reducing operational overhead compared to the other options.
References:
AWS Secrets Manager: https://aws.amazon.com/secrets-manager/
Secrets Manager Rotation:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html IAM for Secrets Manager: https://docs.aws.amazon.com/secretsmanager/latest/userguide/auth-and- access_iam-policies.html

NEW QUESTION: 133
A developer is building an image-processing application that includes an AWS Lambda function. The Lambda function moves images from one AWS service to another AWS service for image processing. For images that are larger than 2 MB, the Lambda function returns the following error: "Task timed out after 3.01 seconds." The developer needs to resolve the error without modifying the Lambda function code.
Which solution will meet these requirements?
A. Increase the Lambda function's timeout value.
B. Configure the Lambda function to not move images that are larger than 2 MB.
C. Request a concurrency quota increase for the Lambda function.
D. Configure provisioned concurrency for the Lambda function.
Answer: A (LEAVE A REPLY)
The error message "Task timed out after 3.01 seconds" indicates that the Lambda invocation exceeded the function's configured timeout setting, which appears to be set to approximately 3 seconds. Larger images (>2 MB) take longer to transfer between services (due to network latency, throughput, and downstream service response time), so the runtime crosses the timeout threshold and Lambda forcibly terminates the execution.
This is a configuration problem, not necessarily a code defect.
To resolve the issue without modifying the function code, the developer should increase the Lambda function's timeout value. Lambda allows configuring the maximum runtime per invocation (up to the service limit). By increasing the timeout to a value that comfortably covers the expected transfer and processing initiation time for larger payloads, the function can complete its work successfully. This directly addresses the failure mode while keeping the existing implementation unchanged.
Option D (provisioned concurrency) reduces cold start latency by keeping execution environments initialized, but it does not extend the allowed runtime for a single invocation. If the function consistently needs more than

3 seconds for larger images, provisioned concurrency will not prevent timeouts. Option C (concurrency quota increase) increases the number of concurrent executions allowed, which can help throughput under load, but again does not affect per-invocation runtime limits. Option B avoids the problem by skipping large images, but it violates the functional need to process them and is not a general solution.
Therefore, the correct solution is A: increase the Lambda timeout so the function has sufficient time to move larger images between AWS services.

NEW QUESTION: 134
A developer is building a microservices-based application by using Python on AWS and several AWS services The developer must use AWS X-Ray The developer views the service map by using the console to view the service dependencies. During testing, the developer notices that some services are missing from the service map What can the developer do to ensure that all services appear in the X-Ray service map?
A. Modify the X-Ray Python agent configuration in each service to increase the sampling rate
B. Instrument the application by using the X-Ray SDK for Python. Install the X-Ray SDK for all the services that the application uses
C. Enable X-Ray data aggregation in Amazon CloudWatch Logs for all the services that the application uses
D. Increase the X-Ray service map timeout value in the X-Ray console
Answer: B (LEAVE A REPLY)
* AWS X-Ray SDK: The primary way to enable X-Ray tracing within applications. The SDK sends data about requests and subsegments to the X-Ray daemon for service map generation.
* Instrumenting All Services: To visualize a complete microservice architecture on the service map, each relevant service must include the X-Ray SDK.
References:
* AWS X-Ray Documentation: https://docs.aws.amazon.com/xray/
* X-Ray SDK for Python: https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-python.html

NEW QUESTION: 135
A developer has created an AWS Lambda function that is written in Python. The Lambda function reads data from objects in Amazon S3 and writes data to an Amazon DynamoDB table. The function is successfully invoked from an S3 event notification when an object is created. However, the function fails when it attempts to write to the DynamoDB table.
What is the MOST likely cause of this issue?
A. The Lambda function's concurrency limit has been exceeded.
B. DynamoDB table requires a global secondary index (GSI) to support writes.
C. The Lambda function does not have IAM permissions to write to DynamoDB.

D. The DynamoDB table is not running in the same Availability Zone as the Lambda function.

Answer: C (LEAVE A REPLY)

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_lambda-access-dynamodb.
html

NEW QUESTION: 136

A company has an application that generates large binary data outside of AWS. The company must encrypt the data before uploading the data to an Amazon S3 bucket.
Which solution will meet this requirement?

A. Use the AWS KMS encrypt command in the AWS CLI.

B. Configure server-side encryption on the Amazon S3 bucket.

C. Use the AWS Encryption SDK to perform client-side encryption of the data.

D. Specify the x-amz-server-side-encryption header when uploading the data to the Amazon S3 bucket.

Answer: C (LEAVE A REPLY)

Comprehensive and Detailed 250 to 300 words of Explanation From AWS Developer Documents:

The key requirement in this scenario is that the data must be encrypted before it is uploaded to Amazon S3, and the data is generated outside of AWS. This explicitly points to a client-side encryption requirement.

AWS documentation distinguishes clearly between server-side encryption and client-side encryption.

Server-side encryption options for Amazon S3-including enabling default bucket encryption or specifying the x-amz-server-side-encryption request header-encrypt the data after it is received by Amazon S3.

Therefore, options B and D do not satisfy the requirement because the plaintext data is transmitted to AWS before encryption occurs.

Option A, using the AWS KMS encrypt command, is not suitable for large binary objects. AWS KMS is designed to encrypt small pieces of data (up to 4 KB). AWS documentation explicitly states that for large payloads, customers should use envelope encryption instead of directly encrypting data with AWS KMS.

The AWS Encryption SDK is specifically designed for client-side encryption of large data objects. It implements envelope encryption, where a data encryption key (DEK) is generated locally to encrypt the data, and the DEK is then encrypted with an AWS KMS key. This approach allows efficient encryption of large binary files while maintaining strong key management and security controls.

AWS documentation recommends the AWS Encryption SDK for applications that need to encrypt data before sending it to AWS services, including Amazon S3. It supports multiple

programming languages, handles key management automatically, and ensures data confidentiality even before transmission.
Therefore, using the AWS Encryption SDK for client-side encryption is the only solution that fully satisfies all requirements.

NEW QUESTION: 137
A development team uses AWS CodeBuild as part of a CI/CD pipeline. The project includes hundreds of unit and integration tests, and total build time continues to increase. The team wants faster feedback and lower overall testing duration without managing additional infrastructure.
Which solution will meet these requirements with the LEAST operational overhead?
A. Configure multiple CodeBuild projects and manually split tests across them.
B. Configure CodeBuild to split tests across multiple parallel compute environments.
C. Run all tests sequentially in a single CodeBuild environment.
D. Use Amazon EC2 instances with a custom test runner to distribute tests.
Answer: (SHOW ANSWER)
AWS CodeBuild supports parallel test execution through batch builds and multiple compute environments.
This feature allows a single CodeBuild project to divide test workloads across multiple isolated environments that run concurrently. Each environment executes a subset of the test suite, significantly reducing total build time.
AWS documentation emphasizes using native CodeBuild capabilities to minimize operational complexity.
Parallel execution eliminates the need to manually manage infrastructure, test orchestration logic, or EC2 fleets. CodeBuild automatically provisions and terminates build environments, scales as needed, and integrates seamlessly with CI/CD pipelines.
Options A and D require manual coordination and infrastructure management, increasing operational overhead. Option C does not address the performance problem.
Therefore, using CodeBuild's parallel compute environments is the most efficient and AWS-recommended approach.

NEW QUESTION: 138

A company has an application that processes audio files for different departments. When audio files are saved to an Amazon S3 bucket, an AWS Lambda function receives an event notification and processes the audio input.
A developer needs to update the solution so that the application can process the audio files for each department independently. The application must publish the audio file location for each department to each department's existing Amazon SQS queue.
Which solution will meet these requirements with no changes to the Lambda function code?
A. Configure the S3 bucket to send the event notifications to an Amazon SNS topic. Subscribe each department's SQS queue to the SNS topic. Configure subscription filter policies.
B. Update the Lambda function to write the file location to a single shared SQS queue. Configure the shared SQS queue to send the file reference to each department's SQS queue.
C. Update the Lambda function to send the file location to each department's SQS queue.
D. Configure the S3 bucket to send the event notifications to each department's SQS queue.
Answer: (SHOW ANSWER)
The key constraint is no changes to the Lambda code, while still fanning out notifications so that each department receives only its relevant file locations in its existing SQS queue.
The cleanest "plumbing-only" pattern is S3 # SNS # SQS with SNS subscription filter policies.
Amazon S3 event notifications can publish events to an SNS topic. SNS then delivers messages to multiple subscribers, including SQS queues. By subscribing each department's SQS queue to the SNS topic, the system can fan out the event to all queues.
To ensure each department receives only its relevant events, the developer can configure SNS subscription filter policies (for example, based on object key prefixes like /deptA/, /deptB/). This routes messages without requiring any change to the Lambda function.
Option D is not ideal because S3 event notifications do not provide the same flexible filtering/routing to multiple SQS queues as SNS filter policies do (and configuring many direct notifications becomes harder to manage).
Options B and C require Lambda code changes, which violates the requirement.
Therefore, use S3 event notifications to SNS, subscribe each department SQS queue, and use filter policies for routing.

NEW QUESTION: 139
A healthcare company uses AWS Amplify to host a patient management system. The system uses Amazon API Gateway to expose RESTful APIs. The backend logic of the system is handled by AWS Lambda functions.

One of the Lambda functions receives patient data that includes personally identifiable information (PII). The Lambda function sends the patient data to an Amazon DynamoDB table. The company must encrypt all patient data at rest and in transit before the data is stored in DynamoDB.
Which solution will meet these requirements?
A. Configure the Lambda function to use AWS KMS keys with the AWS Database Encryption SDK to encrypt the patient data before sending the data to DynamoDB.
B. Use AWS managed AWS KMS keys to encrypt the data in the DynamoDB table.
C. Configure a DynamoDB stream on the table to invoke a Lambda function. Configure the Lambda function use an AWS KMS key to encrypt the DynamoDB table and to update the table.
D. Use an AWS Step Functions workflow to transfer the data to an Amazon SQS queue. Configure a Lambda function to encrypt the data in the queue before sending the data to the DynamoDB table.
Answer: (SHOW ANSWER)
DynamoDB already supports encryption at rest by default (with AWS owned keys or AWS managed
/customer managed KMS keys). However, the requirement here is stronger and explicit: patient data must be encrypted before it is stored in DynamoDB, meaning the encryption must occur client-side / application-side so that the data remains encrypted as it traverses the stack and is still encrypted when written to the table. This is often required for highly sensitive PII/PHI where administrators or downstream systems should not see plaintext.
Option A meets this requirement by encrypting the payload inside the Lambda function before making the PutItem call to DynamoDB, using KMS-backed envelope encryption via the AWS Encryption SDK (the option names it "Database Encryption SDK," but the intent is client-side encryption). The encrypted fields remain ciphertext in transit to DynamoDB and at rest inside DynamoDB, satisfying both "in transit" and "at rest" with encryption happening prior to storage.
Option B only ensures DynamoDB encryption at rest at the service level, but the data would still be plaintext in the item attributes as seen by the application and anyone with DynamoDB read permissions (even though stored encrypted on disk). It does not guarantee "encrypted before stored." Option C encrypting after the write via streams is too late: plaintext would already be stored (and potentially accessed) before post-processing.
Option D adds unnecessary workflow complexity. Encrypting in a queue still requires encryption before DynamoDB, but it does not inherently simplify compared to encrypting directly in the Lambda that writes to DynamoDB.
Therefore, client-side encryption in Lambda using KMS-backed encryption is the correct solution.

NEW QUESTION: 140

A company is concerned that a malicious user could deploy unauthorized changes to the code for an AWS Lambda function. What can a developer do to ensure that only trusted code is deployed to Lambda?

A. Turn on the trusted code option in AWS CodeDeploy. Add the CodeDeploy digital certificate to the Lambda package before deploying the package to Lambda.

B. Define the code signing configuration in the Lambda console. Use AWS Signer to digitally sign the Lambda package before deploying the package to Lambda.

C. Link Lambda to AWS KMS in the Lambda console. Use AWS KMS to digitally sign the Lambda package before deploying the package to Lambda.

D. Set the KmsKeyArn property of the Lambda function to the Amazon Resource Name (ARN) of a trusted key before deploying the package to Lambda.

Answer: B (LEAVE A REPLY)

Requirement Summary:

Prevent unauthorized code changes in AWS Lambda

Ensure only trusted code is deployed

AWS Lambda supports Code Signing:

You can configure code signing in Lambda using AWS Signer

Packages must be digitally signed and verified against the signing profile Rejects unauthorized/modified packages automatically Evaluate Options:

A). Trusted code option in CodeDeploy

No such feature exists for Lambda

CodeDeploy is more for EC2/On-Prem/Containers, not Lambda code signing

B). Define code signing config + use AWS Signer

This is exactly how AWS enforces trusted code deployment

Attach a code signing configuration to the Lambda function

Use AWS Signer to digitally sign deployment packages

C). Link to KMS to sign code

KMS is not used to sign Lambda packages

KMS is for data encryption, not application code integrity

D). Set KmsKeyArn

This configures data encryption, not code signing

Lambda code signing: https://docs.aws.amazon.com/lambda/latest/dg/configuration-codesigning.html AWS Signer overview: https://docs.aws.amazon.com/signer/latest/developerguide/what-is-signer.html

NEW QUESTION: 141

A company needs to harden its container images before the images are in a running state. The company's application uses Amazon Elastic Container Registry (Amazon ECR) as an image registry. Amazon Elastic Kubernetes Service (Amazon EKS) for compute, and an AWS CodePipeline pipeline that orchestrates a continuous integration and continuous delivery (CI/CD) workflow.

Dynamic application security testing occurs in the final stage of the pipeline after a new image is deployed to a development namespace in the EKS cluster. A developer needs to place an analysis stage before this deployment to analyze the container image earlier in the CI/CD pipeline.
Which solution will meet these requirements with the MOST operational efficiency?
A. Build the container image and run the docker scan command locally. Mitigate any findings before pushing changes to the source code repository. Write a pre-commit hook that enforces the use of this workflow before commit.
B. Create a new CodePipeline stage that occurs after the container image is built. Configure ECR basic image scanning to scan on image push. Use an AWS Lambda function as the action provider. Configure the Lambda function to check the scan results and to fail the pipeline if there are findings.
C. Create a new CodePipeline stage that occurs after source code has been retrieved from its repository.
Run a security scanner on the latest revision of the source code. Fail the pipeline if there are findings.
D. Add an action to the deployment stage of the pipeline so that the action occurs before the deployment to the EKS cluster. Configure ECR basic image scanning to scan on image push. Use an AWS Lambda function as the action provider. Configure the Lambda function to check the scan results and to fail the pipeline if there are findings.
Answer: B (LEAVE A REPLY)
The solution that will meet the requirements with the most operational efficiency is to create a new CodePipeline stage that occurs after the container image is built. Configure ECR basic image scanning to scan on image push. Use an AWS Lambda function as the action provider. Configure the Lambda function to check the scan results and to fail the pipeline if there are findings. This way, the container image is analyzed earlier in the CI/CD pipeline and any vulnerabilities are detected and reported before deploying to the EKS cluster. The other options either delay the analysis until after deployment, which increases the risk of exposing insecure images, or perform analysis on the source code instead of the container image, which may not capture all the dependencies and configurations that affect the security posture of the image.
Reference: Amazon ECR image scanning

NEW QUESTION: 142
A company has a social media application that receives large amounts of traffic User posts and interactions are continuously updated in an Amazon RDS database The data changes frequently, and the data types can be complex The application must serve read requests with minimal latency The application's current architecture struggles to deliver these rapid data updates efficiently The company needs a solution to improve the application's performance.
Which solution will meet these requirements'?

A. Use Amazon DynamoDB Accelerator (DAX) in front of the RDS database to provide a caching layer for the high volume of rapidly changing data
B. Set up Amazon S3 Transfer Acceleration on the RDS database to enhance the speed of data transfer from the databases to the application.
C. Add an Amazon CloudFront distribution in front of the RDS database to provide a caching layer for the high volume of rapidly changing data
D. Create an Amazon ElastiCache for Redis cluster. Update the application code to use a write-through caching strategy and read the data from Redis.
Answer: D (LEAVE A REPLY)
Amazon ElastiCache for Redis: An in-memory data store known for extremely low latency, ideal for caching frequently accessed, complex data.
Write-Through Caching: Ensures that data is always consistent between the cache and the database. Writes go to both Redis and RDS.
Performance Gains: Redis handles reads with minimal latency, offloading the RDS database and improving the application's responsiveness.
References:
Amazon ElastiCache for Redis Documentation:
https://docs.aws.amazon.com/AmazonElastiCache/latest/red- ug/ Caching Strategies:
https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Strategies.html

NEW QUESTION: 143
A developer migrated a legacy application to an AWS Lambda function. The function uses a third-party service to pull data with a series of API calls at the end of each month. The function than processes the data to generate the monthly reports. The function has Been working with no issues so far.
The third-party service recently issued a restriction to allow a feed number to API calls each minute and each day. If the API calls exceed the limit tor each minute or each day, then the service will produce errors. The API also provides the minute limit and daily limit in the response header. This restriction might extend the overall process to multiple days because the process is consuming more API calls than the available limit.
What is the MOST operationally efficient way to refactor the server less application to accommodate this change?
A. Use an AWS Step Functions State machine to monitor API failures. Use the Wait state to delay calling the Lambda function.
B. Use an Amazon Simple Queue Service (Amazon SQS) queue to hold the API calls. Configure the Lambda function to poll the queue within the API threshold limits.
C. Use an Amazon CloudWatch Logs metric to count the number of API calls. Configure an Amazon CloudWatch alarm flat slops the currently running instance of the Lambda function when the metric exceeds the API threshold limits.
D. Use Amazon Kinesis Data Firehose to batch me API calls and deliver them to an Amazon S3 bucket win an event notification to invoke the Lambda function.

Answer: (SHOW ANSWER)
The solution that will meet the requirements is to use an AWS Step Functions state machine to monitor API failures. Use the Wait state to delay calling the Lambda function. This way, the developer can refactor the serverless application to accommodate the change in a way that is automated and scalable. The developer can use Step Functions to orchestrate the Lambda function and handle any errors or retries. The developer can also use the Wait state to pause the execution for a specified duration or until a specified timestamp, which can help avoid exceeding the API limits. The other options either involve using additional services that are not necessary or appropriate for this scenario, or do not address the issue of API failures.
Reference: AWS Step Functions Wait state

NEW QUESTION: 144
A developer is deploying an application on Amazon EC2 instances that run in Account A. The application needs to read data from an existing Amazon Kinesis data stream in Account B.
Which actions should the developer take to provide the application with access to the stream? (Select TWO.)
A. Update the instance profile role in Account A with stream read permissions.
B. Create an IAM role with stream read permissions in Account B.
C. Add a trust policy to the instance profile role and IAM role in Account B to allow the instance profile role to assume the IAM role.
D. Add a trust policy to the instance profile role and IAM role in Account B to allow reads from the stream.
E. Add a resource-based policy in Account B to allow read access from the instance profile role.
Answer: (SHOW ANSWER)
For cross-account access from EC2 in Account A to a Kinesis data stream in Account B, the recommended secure pattern is to use STS AssumeRole into a role in the resource-owning account (Account B). This ensures Account B retains control over what permissions are granted to external principals and allows Account A workloads to obtain temporary credentials scoped to the required actions.
First, create an IAM role in Account B that has the necessary Kinesis read permissions (such as kinesis:
GetRecords, kinesis:GetShardIterator, kinesis:DescribeStream, and related read actions as required). That corresponds to option B. This role represents the permission boundary controlled by Account B for accessing its stream.
Second, configure the role's trust policy in Account B to allow the instance profile role (or another IAM principal) from Account A to assume it. In practice, the trust relationship is defined on the Account B role and specifies the Account A role as the trusted principal,

enabling sts:AssumeRole. This corresponds to option C (the intent is "allow the instance profile role to assume the IAM role in Account B").
Option A alone is insufficient because permissions in Account A do not grant access to resources owned by Account B without an explicit cross-account authorization path. Option D is incorrect because trust policies do not "allow reads from the stream"; they allow principals to assume roles, and permissions policies allow service actions. Option E (resource-based policy) is not the primary mechanism for Kinesis Data Streams cross-account access in this scenario compared with the standard role assumption model; the secure and common approach is to assume a role in the owning account.
Therefore, the correct actions are B (create the role with read permissions in Account B) and C (configure trust to allow Account A's instance role to assume it).

NEW QUESTION: 145
A developer needs to perform geographic load testing of an API. The developer must deploy resources to multiple AWS Regions to support the load testing of the API.
How can the developer meet these requirements without additional application code?
A. Create and deploy an AWS Lambda function in each desired Region. Configure the Lambda function to create a stack from an AWS CloudFormation template in that Region when the function is invoked.
B. Create an AWS CloudFormation template that defines the load test resources. Use the AWS CLI create-stack-set command to create a stack set in the desired Regions.
C. Create an AWS Systems Manager document that defines the resources. Use the document to create the resources in the desired Regions.
D. Create an AWS CloudFormation template that defines the load test resources. Use the AWS CLI deploy command to create a stack from the template in each Region.
Answer: B (LEAVE A REPLY)
AWS CloudFormation is a service that allows developers to model and provision AWS resources using templates. A CloudFormation template can define the load test resources, such as EC2 instances, load balancers, and Auto Scaling groups. A CloudFormation stack set is a collection of stacks that can be created and managed from a single template in multiple Regions and accounts. The AWS CLI create-stack-set command can be used to create a stack set from a template and specify the Regions where the stacks should be created. Reference: Working with AWS CloudFormation stack sets

NEW QUESTION: 146
A developer at a company needs to create a small application that makes the same API call once each day at a designated time. The company does not have infrastructure in the AWS Cloud yet, but the company wants to implement this functionality on AWS.
Which solution meets these requirements in the MOST operationally efficient manner?
A. Use a Kubernetes cron job that runs on Amazon Elastic Kubernetes Service (Amazon EKS).

B. Use an Amazon Linux crontab scheduled job that runs on Amazon EC2.
C. Use an AWS Lambda function that is invoked by an Amazon EventBridge scheduled event.
D. Use an AWS Batch job that is submitted to an AWS Batch job queue.
Answer: C (LEAVE A REPLY)
The correct answer is C. Use an AWS Lambda function that is invoked by an Amazon EventBridge scheduled event.
C: Use an AWS Lambda function that is invoked by an Amazon EventBridge scheduled event. This is correct.
AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging1. Amazon EventBridge is a serverless event bus service that enables you to connect your applications with data from a variety of sources2. EventBridge can create rules that run on a schedule, either at regular intervals or at specific times and dates, and invoke targets such as Lambda functions3. This solution meets the requirements of creating a small application that makes the same API call once each day at a designated time, without requiring any infrastructure in the AWS Cloud or any operational overhead.
A; Use a Kubernetes cron job that runs on Amazon Elastic Kubernetes Service (Amazon EKS). This is incorrect. Amazon EKS is a fully managed Kubernetes service that allows you to run containerized applications on AWS4. Kubernetes cron jobs are tasks that run periodically on a given schedule5. This solution could meet the functional requirements of creating a small application that makes the same API call once each day at a designated time, but it would not be the most operationally efficient manner. The company would need to provision and manage an EKS cluster, which would incur additional costs and complexity.
B: Use an Amazon Linux crontab scheduled job that runs on Amazon EC2. This is incorrect. Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud6. Crontab is a Linux utility that allows you to schedule commands or scripts to run automatically at a specified time or date7. This solution could meet the functional requirements of creating a small application that makes the same API call once each day at a designated time, but it would not be the most operationally efficient manner. The company would need to provision and manage an EC2 instance, which would incur additional costs and complexity.
D: Use an AWS Batch job that is submitted to an AWS Batch job queue. This is incorrect. AWS Batch enables you to run batch computing workloads on the AWS Cloud8. Batch jobs are units of work that can be submitted to job queues, where they are executed in parallel or sequentially on compute environments9. This solution could meet the functional requirements of creating a small application that makes the same API call once each day at a designated time, but it would not be the most operationally efficient manner. The

company would need to configure and manage an AWS Batch environment, which would incur additional costs and complexity.
References:
* 1: What is AWS Lambda? - AWS Lambda
* 2: What is Amazon EventBridge? - Amazon EventBridge
* 3: Creating an Amazon EventBridge rule that runs on a schedule - Amazon EventBridge
* 4: What is Amazon EKS? - Amazon EKS
* 5: CronJob - Kubernetes
* 6: What is Amazon EC2? - Amazon EC2
* 7: Crontab in Linux with 20 Useful Examples to Schedule Jobs - Tecmint
* 8: What is AWS Batch? - AWS Batch
* 9: Jobs - AWS Batch

NEW QUESTION: 147
A company has an application that runs as a series of AWS Lambda functions. Each Lambda function receives data from an Amazon Simple Notification Service (Amazon SNS) topic and writes the data to an Amazon Aurora DB instance.
To comply with an information security policy, the company must ensure that the Lambda functions all use a single securely encrypted database connection string to access Aurora.
Which solution will meet these requirements'?
A. Use IAM database authentication for Aurora to enable secure database connections for ail the Lambda functions.
B. Store the credentials and read the credentials from an encrypted Amazon RDS DB instance.
C. Store the credentials in AWS Systems Manager Parameter Store as a secure string parameter.
D. Use Lambda environment variables with a shared AWS Key Management Service (AWS KMS) key for encryption.
Answer: A (LEAVE A REPLY)
This solution will meet the requirements by using IAM database authentication for Aurora, which enables using IAM roles or users to authenticate with Aurora databases instead of using passwords or other secrets.
The developer can use IAM database authentication for Aurora to enable secure database connections for all the Lambda functions that access Aurora DB instance. The developer can create an IAM role with permission to connect to Aurora DB instance and attach it to each Lambda function. The developer can also configure Aurora DB instance to use IAM database authentication and enable encryption in transit using SSL certificates. This way, the Lambda functions can use a single securely encrypted database connection string to access Aurora without needing any secrets or passwords. Option B is not optimal because it will store the credentials and read them from an encrypted Amazon RDS DB instance, which may introduce additional costs and complexity for managing and accessing another

RDS DB instance. Option C is not optimal because it will store the credentials in AWS Systems Manager Parameter Store as a secure string parameter, which may require additional steps or permissions to retrieve and decrypt the credentials from Parameter Store.
Option D is not optimal because it will use Lambda environment variables with a shared AWS Key Management Service (AWS KMS) key for encryption, which may not be secure or scalable as environment variables are stored as plain text unless encrypted with AWS KMS.
References: [IAM Database Authentication for MySQL and PostgreSQL], [Using SSL/TLS to Encrypt a Connection to a DB Instance]

NEW QUESTION: 148
A developer has written an AWS Lambda function. The function is CPU-bound. The developer wants to ensure that the function returns responses quickly.
How can the developer improve the function's performance?
A. Increase the function's CPU core count.
B. Increase the function's memory.
C. Increase the function's reserved concurrency.
D. Increase the function's timeout.
Answer: B (LEAVE A REPLY)
The amount of memory you allocate to your Lambda function also determines how much CPU and network bandwidth it gets. Increasing the memory size can improve the performance of CPU-bound functions by giving them more CPU power. The CPU allocation is proportional to the memory allocation, so a function with 1 GB of memory has twice the CPU power of a function with 512 MB of memory. Reference: AWS Lambda execution environment

NEW QUESTION: 149
A team of developed is using an AWS CodePipeline pipeline as a continuous integration and continuous delivery (CI/CD) mechanism for a web application. A developer has written unit tests to programmatically test the functionality of the application code. The unit tests produce a test report that shows the results of each individual check. The developer now wants to run these tests automatically during the CI/CD process.
A. Write a Git pre-commit hook that runs the test before every commit. Ensure that each developer who is working on the project has the pre-commit hook instated locally. Review the test report and resolve any issues before pushing changes to AWS CodeCommit.
B. Add a new stage to the pipeline. Use AWS CodeBuild as the provider. Add the new stage after the stage that deploys code revisions to the test environment. Write a buildspec that fails the CodeBuild stage if any test does not pass. Use the test reports feature of Codebuild to integrate the report with the CodoBuild console. View the test results in CodeBuild Resolve any issues.

C. Add a new stage to the pipeline. Use AWS CodeBuild at the provider. Add the new stage before the stage that deploys code revisions to the test environment. Write a buildspec that fails the CodeBuild stage it any test does not pass. Use the test reports feature of CodeBuild to integrate the report with the CodeBuild console. View the test results in codeBuild Resolve any issues.
D. Add a new stage to the pipeline. Use Jenkins as the provider. Configure CodePipeline to use Jenkins to run the unit tests. Write a Jenkinsfile that fails the stage if any test does not pass. Use the test report plugin for Jenkins to integrate the repot with the Jenkins dashboard. View the test results in Jenkins.Resolve any issues.
Answer: C (LEAVE A REPLY)
The solution that will meet the requirements is to add a new stage to the pipeline. Use AWS CodeBuild as the provider. Add the new stage before the stage that deploys code revisions to the test environment. Write a buildspec that fails the CodeBuild stage if any test does not pass. Use the test reports feature of CodeBuild to integrate the report with the CodeBuild console. View the test results in CodeBuild. Resolve any issues. This way, the developer can run the unit tests automatically during the CI/CD process and catch any bugs before deploying to the test environment. The developer can also use the test reports feature of CodeBuild to view and analyze the test results in a graphical interface. The other options either involve running the tests manually, running them after deployment, or using a different provider that requires additional configuration and integration.
Reference: Test reports for CodeBuild

NEW QUESTION: 150
A developer is writing a serverless application that requires an AWS Lambda function to be invoked every 10 minutes.
What is an automated and serverless way to invoke the function?
A. Deploy an Amazon EC2 instance based on Linux, and edit its /etc/confab file by adding a command to periodically invoke the lambda function
B. Configure an environment variable named PERIOD for the Lambda function. Set the value to 600.
C. Create an Amazon EventBridge rule that runs on a regular schedule to invoke the Lambda function.
D. Create an Amazon Simple Notification Service (Amazon SNS) topic that has a subscription to the Lambda function with a 600-second timer.
Answer: C (LEAVE A REPLY)
The solution that will meet the requirements is to create an Amazon EventBridge rule that runs on a regular schedule to invoke the Lambda function. This way, the developer can use an automated and serverless way to invoke the function every 10 minutes. The developer can also use a cron expression or a rate expression to specify the schedule for the rule. The other options either involve using an Amazon EC2 instance, which is not

serverless, or using environment variables or query parameters, which do not trigger the function.
Reference: Schedule AWS Lambda functions using EventBridge

NEW QUESTION: 151
A developer is building an application that uses Amazon DynamoDB. The developer wants to retrieve multiple specific items from the database with a single API call. Which DynamoDB API call will meet these requirements with the MINIMUM impact on the database?
A. GetItem
B. BatchGetItem
C. Query
D. Scan
Answer: B (LEAVE A REPLY)

NEW QUESTION: 152
A developer maintains a serverless application that uses an Amazon API Gateway REST API to invoke an AWS Lambda function by using a non-proxy integration. The Lambda function returns data, which is stored in Amazon DynamoDB.
Several application users begin to receive intermittent errors from the API. The developer examines Amazon CloudWatch Logs for the Lambda function and discovers several ProvisionedThroughputExceededException errors.
The developer needs to resolve the errors and ensure that the errors do not reoccur.
A. Use provisioned capacity mode for the DynamoDB table, and assign sufficient capacity units.
Configure the Lambda function to retry requests with exponential backoff.
B. Update the REST API to send requests on an Amazon SQS queue. Configure the Lambda function to process requests from the queue.
C. Configure a usage plan for the REST API.
D. Update the REST API to invoke the Lambda function asynchronously.
Answer: A (LEAVE A REPLY)
Comprehensive and Detailed Step-by-Step Explanation:

* Option A: Provisioned Capacity with Exponential Backoff:
* Using provisioned capacity ensures sufficient throughput for the DynamoDB table.
* Configuring the Lambda function to implement exponential backoff retries reduces the chance of exceeding capacity during peak usage.
* This combination addresses the root cause (ProvisionedThroughputExceededException) and prevents errors without overprovisioning.
* Why Other Options Are Incorrect:
* Option B:Using SQS adds unnecessary latency and complexity. The issue lies in DynamoDB throughput, not request management.
* Option C:A usage plan for the API does not address throughput issues in DynamoDB.
* Option D:Invoking the Lambda function asynchronously does not resolve the DynamoDB capacity issue and might lead to delayed processing.
References:
* DynamoDB Provisioned Throughput Documentation

NEW QUESTION: 153
A company runs a serverless application that uses several AWS Lambda functions. The existing Lambda functions run in a VPC. The Lambda functions query public APIs successfully.
To add a new feature to the application, a developer creates a new Lambda function to query external public APIs. The new Lambda function must store aggregated results in an Amazon RDS database that is in a private subnet of the VPC. The developer configures VPC access for the new Lambda function and sets up a working connection to the RDS database. The requests that the new Lambda function makes to the external APIs fail. However, requests from the developer's local workstation to the same APIs are successful.
Which solution will meet this requirement?
A. Provision an elastic network interface for the new Lambda function.
B. Provision a NAT gateway in a public subnet in the VPC.
C. Provision an outbound rule for the new Lambda function's security group to grant internet access.
D. Provision a gateway VPC endpoint in a public subnet in the VPC.
Answer: (SHOW ANSWER)
When a Lambda function is configured to run inside a VPC, it receives ENIs in the selected subnets and uses those subnets' routing to reach other networks. If the function is placed in private subnets (common when it needs to access an RDS database in private subnets), it typically does not have a direct route to the internet.
As a result, outbound calls to public external APIs will fail unless the VPC provides a controlled egress path.
The standard AWS networking pattern for private-subnet internet access is a NAT gateway (or NAT instance) deployed in a public subnet, with a route from the private subnet(s) to the NAT gateway (0.0.0.0/0

NAT). The NAT gateway then uses an internet gateway attached to the VPC to reach the public internet while keeping the Lambda function (and the RDS database) in private address space. This resolves the connectivity issue while maintaining the security posture of private subnets.
Option A is not the solution: Lambda automatically provisions ENIs for VPC-enabled functions; manually provisioning an ENI does not provide internet access. Option C (security group outbound rules) is necessary but not sufficient-security groups control allowed traffic, but without a proper route to the internet (via NAT), the packets still cannot reach external endpoints. Option D (gateway VPC endpoint) is for AWS services like S3 and DynamoDB (gateway endpoints), not for arbitrary public internet APIs, and "in a public subnet" is not how gateway endpoints are used; they are associated with route tables.
Therefore, the correct solution is B: add a NAT gateway in a public subnet and update private subnet route tables so the new Lambda function can reach external public APIs while still accessing the private RDS database.

NEW QUESTION: 154
A company has an AWS Step Functions state machine named myStateMachine. The company configured a service role for Step Functions. The developer must ensure that only the myStateMachine state machine can assume the service role.
Which statement should the developer add to the trust policy to meet this requirement?
A. "Condition": { "ArnLike": { "aws:SourceArn":"urn:aws:states:ap-south-1:111111111111:stateMachine:myStateMachine" } }
B. "Condition": { "ArnLike": { "aws:SourceArn":"arn:aws:states:ap-south-1:*:stateMachine:myStateMachine" } }
C. "Condition": { "StringEquals": { "aws:SourceAccount": "111111111111" } }
D. "Condition": { "StringNotEquals": { "aws:SourceArn":"arn:aws:states:ap-south-1:111111111111:stateMachine:myStateMachine" } }
Answer: (SHOW ANSWER)
* Why Option A is Correct:
* The ArnLike condition with the specific ARN for myStateMachine ensures that only this state machine can assume the role. The format urn:aws:states is correct for specifying Step Functions resources.
* Why Other Options are Incorrect:
* Option B: Wildcards (*) in the ARN allow more resources to assume the role, which violates the requirement.
* Option C: This condition restricts the account but not the specific state machine.
* Option D: A StringNotEquals condition is used to deny specific values, which does not ensure exclusivity for the desired state machine.
* AWS Documentation References:

* IAM Trust Policies for Step Functions

NEW QUESTION: 155
A company needs to set up secure database credentials for all its AWS Cloud resources. The company's resources include Amazon RDS DB instances Amazon DocumentDB clusters and Amazon Aurora DB instances. The company's security policy mandates that database credentials be encrypted at rest and rotated at a regular interval.
Which solution will meet these requirements MOST securely?
A. Set up IAM database authentication for token-based access. Generate user tokens to provide centralized access to RDS DB instances. Amazon DocumentDB clusters and Aurora DB instances.
B. Create parameters for the database credentials in AWS Systems Manager Parameter Store Set the Type parameter to Secure Sting. Set up automatic rotation on the parameters.
C. Store the database access credentials as an encrypted Amazon S3 object in an S3 bucket Block all public access on the S3 bucket. Use S3 server-side encryption to set up automatic rotation on the encryption key.
D. Create an AWS Lambda function by using the SecretsManagerRotationTemplate template in the AWS Secrets Manager console. Create secrets for the database credentials in Secrets Manager Set up secrets rotation on a schedule.
Answer: D (LEAVE A REPLY)
This solution will meet the requirements by using AWS Secrets Manager, which is a service that helps protect secrets such as database credentials by encrypting them with AWS Key Management Service (AWS KMS) and enabling automatic rotation of secrets. The developer can create an AWS Lambda function by using the SecretsManagerRotationTemplate template in the AWS Secrets Manager console, which provides a sample code for rotating secrets for RDS DB instances, Amazon DocumentDB clusters, and Amazon Aurora DB instances. The developer can also create secrets for the database credentials in Secrets Manager, which encrypts them at rest and provides secure access to them. The developer can set up secrets rotation on a schedule, which changes the database credentials periodically according to a specified interval or event.
Option A is not optimal because it will set up IAM database authentication for token-based access, which may not be compatible with all database engines and may require additional configuration and management of IAM roles or users. Option B is not optimal because it will create parameters for the database credentials in AWS Systems Manager Parameter Store, which does not support automatic rotation of secrets. Option C is not optimal because it will store the database access credentials as an encrypted Amazon S3 object in an S3 bucket, which may introduce additional costs and complexity for accessing and securing the data.
References: [AWS Secrets Manager], [Rotating Your AWS Secrets Manager Secrets]

NEW QUESTION: 156
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 (LEAVE A REPLY)
The workload is a classic "hot key / hot dataset" pattern: many reads repeatedly target a small subset of items.
The best way to improve performance and reduce latency is to add a caching layer so the application does not hit DynamoDB for every read.
Option B is purpose-built for DynamoDB read acceleration: DynamoDB Accelerator (DAX) is an in-memory cache that sits in front of DynamoDB and is API-compatible for many DynamoDB operations. DAX can dramatically reduce read latency (often to microseconds) for frequently accessed items and offload read traffic from the table.
Option A can also help: ElastiCache (Redis/Memcached) can be used as an application-managed cache. This is useful when the application wants more control over caching strategy, TTLs, and non-DynamoDB data caching. For ECS applications, ElastiCache is a common high-performance caching choice.
Why not the others:
* C (strongly consistent reads) typically increases latency and capacity consumption; it does not improve performance for repeated reads.
* D (increase read capacity) can reduce throttling, but it does not reduce latency as effectively as caching and can be more expensive for hot-read patterns.
* E (adaptive capacity) helps DynamoDB handle uneven workloads, but it is not a direct performance boost like caching for repeated reads of a small dataset.
Therefore, adding caching via ElastiCache and/or DAX best improves performance. With "select two," A and B are correct.

NEW QUESTION: 157
A gaming application stores scores for players in an Amazon DynamoDB table that has four attributes:
user_id, user_name, user_score, and user_rank. The users are allowed to update their names only. A user is authenticated by web identity federation.

Which set of conditions should be added in the policy attached to the role for the dynamodb:PutItem API call?

A. "Condition": {"ForAllValues:StringEquals": {"dynamodb:LeadingKeys": ["${www.amazon.com:
user_id}"],"dynamodb:Attributes": ["user_name"]}}

B. "Condition": {"ForAllValues:StringEquals": {"dynamodb:LeadingKeys": ["${www.amazon.com:
user_name}"],"dynamodb:Attributes": ["user_id"]}}

C. "Condition": {"ForAllValues:StringEquals": {"dynamodb:LeadingKeys": ["${www.amazon.com:
user_id}"],"dynamodb:Attributes": ["user_name", "user_id"]}}

D. "Condition": {"ForAllValues:StringEquals": {"dynamodb:LeadingKeys": ["${www.amazon.com:user_name}"],"dynamodb:Attributes": ["username", "userid"]}}

Answer: (SHOW ANSWER)

The correct policy condition ensures that:

The LeadingKeys condition restricts operations to the authenticated user's user_id.

The Attributes condition limits the updatable attributes to user_name.

Explanation of Choices:

Option A: Correctly enforces both the key restriction (dynamodb:LeadingKeys) and ensures only the user_name attribute can be updated.

Option B, C, D: Use incorrect conditions, such as referencing user_name in the LeadingKeys or including other attributes like user_id in updatable fields.

Reference:AWS DynamoDB Condition Keys Documentation

NEW QUESTION: 158

A developer is creating an AWS Lambda function in VPC mode An Amazon S3 event will invoke the Lambda function when an object is uploaded into an S3 bucket The Lambda function will process the object and produce some analytic results that will be recorded into a file Each processed object will also generate a log entry that will be recorded into a file. Other Lambda functions. AWS services, and on-premises resources must have access to the result files and log file. Each log entry must also be appended to the same shared log file. The developer needs a solution that can share files and append results into an existing file.

Which solution should the developer use to meet these requirements?

A. Create an Amazon Elastic File System (Amazon EFS) file system. Mount the EFS file system in Lambda. Store the result files and log file in the mount point. Append the log entries to the log file.

B. Create an Amazon Elastic Block Store (Amazon EBS) Multi-Attach enabled volume Attach the EBS volume to all Lambda functions. Update the Lambda function code to download the log file, append the log entries, and upload the modified log file to Amazon EBS

C. Create a reference to the /tmp local directory. Store the result files and log file by using the directory reference. Append the log entry to the log file.
D. Create a reference to the /opt storage directory Store the result files and log file by using the directory reference Append the log entry to the log file
Answer: A (LEAVE A REPLY)
Amazon EFS: A network file system (NFS) providing shared, scalable storage across multiple Lambda functions and other AWS resources.
Lambda Mounting: EFS file systems can be mounted within Lambda functions to access a shared storage space.
Log Appending: EFS supports appending data to existing files, making it ideal for the log file scenario.
References:
Amazon EFS Documentation: https://docs.aws.amazon.com/efs/
Using Amazon EFS with AWS Lambda:
https://docs.aws.amazon.com/lambda/latest/dg/services-efs.html

NEW QUESTION: 159
An online food company provides an Amazon API Gateway HTTP API 1o receive orders for partners. The API is integrated with an AWS Lambda function. The Lambda function stores the orders in an Amazon DynamoDB table.
The company expects to onboard additional partners Some to me panthers require additional Lambda function to receive orders. The company has created an Amazon S3 bucket. The company needs 10 store all orders and updates m the S3 bucket for future analysis How can the developer ensure that an orders and updates are stored to Amazon S3 with the LEAST development effort?
A. Create a new Lambda function and a new API Gateway API endpoint. Configure the new Lambda function to write to the S3 bucket. Modify the original Lambda function to post updates to the new API endpoint.
B. Use Amazon Kinesis Data Streams to create a new data stream. Modify the Lambda function to publish orders to the oats stream Configure in data stream to write to the S3 bucket.
C. Enable DynamoDB Streams on me DynamoOB table. Create a new lambda function. Associate the stream's Amazon Resource Name (ARN) with the Lambda Function Configure the Lambda function to write to the S3 bucket as records appear in the table's stream.
D. Modify the Lambda function to punish to a new Amazon. Simple Lambda function receives orders.Subscribe a new Lambda function to the topic. Configure the new Lambda function to write to the S3 bucket as updates come through the topic.
Answer: C (LEAVE A REPLY)
This solution will ensure that all orders and updates are stored to Amazon S3 with the least development effort because it uses DynamoDB Streams to capture changes in the

DynamoDB table and trigger a Lambda function to write those changes to the S3 bucket. This way, the original Lambda function and API Gateway API endpoint do not need to be modified, and no additional services are required. Option A is not optimal because it will require more development effort to create a new Lambda function and a new API Gateway API endpoint, and to modify the original Lambda function to post updates to the new API endpoint. Option B is not optimal because it will introduce additional costs and complexity to use Amazon Kinesis Data Streams to create a new data stream, and to modify the Lambda function to publish orders to the data stream. Option D is not optimal because it will require more development effort to modify the Lambda function to publish to a new Amazon SNS topic, and to create and subscribe a new Lambda function to the topic. References: Using DynamoDB Streams, Using AWS Lambda with Amazon S3

NEW QUESTION: 160
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 CodeCommt project. Add the repository package's build and test commands to the protects buildspec
B. Create an AWS CodeBuid project. Add the repository package's build and test commands to the projects buildspec
C. Create an AWS CodeDeploy protect. 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 attract as the actions input artifact.
E. Add a new stage to the pipeline alter the source stage. Add an action to the new stage. Speedy the newly created protect as the action provider. Specify the source artifact as the action's input artifact.
Answer: B,E (LEAVE A REPLY)
This solution will ensure that the repository package's unit tests run in the new deployment environment with the least overhead because it uses AWS CodeBuild to build and test the code in a fully managed service, and AWS CodePipeline to orchestrate the deployment stages and actions. Option A is not optimal because it will use AWS CodeCommit instead of AWS CodeBuild, which is a source control service, not a build and test service. Option C is not optimal because it will use AWS CodeDeploy instead of AWS CodeBuild, which is a deployment service, not a build and test service. Option D is not optimal because it will add an action to the source stage instead of creating a new stage, which will not follow the best practice of separating different deployment phases.

References: AWS CodeBuild, AWS CodePipeline

NEW QUESTION: 161
A company is offering APIs as a service over the internet to provide unauthenticated read access to statistical information that is updated daily. The company uses Amazon API Gateway and AWS Lambda to develop the APIs. The service has become popular, and the company wants to enhance the responsiveness of the APIs.
Which action can help the company achieve this goal?
A. Enable API caching in API Gateway.
B. Configure API Gateway to use an interface VPC endpoint.
C. Enable cross-origin resource sharing (CORS) for the APIs.
D. Configure usage plans and API keys in API Gateway.
Answer: (SHOW ANSWER)
Amazon API Gateway is a service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. The developer can enable API caching in API Gateway to cache responses from the backend integration point for a specified time-to-live (TTL) period. This can improve the responsiveness of the APIs by reducing the number of calls made to the backend service.
References:
* [What Is Amazon API Gateway? - Amazon API Gateway]
* [Enable API Caching to Enhance Responsiveness - Amazon API Gateway]

NEW QUESTION: 162
A developer maintains a critical business application that uses Amazon DynamoDB as the primary data store The DynamoDB table contains millions of documents and receives 30-60 requests each minute The developer needs to perform processing in near-real time on the documents when they are added or updated in the DynamoDB table How can the developer implement this feature with the LEAST amount of change to the existing application code?
A. Set up a cron job on an Amazon EC2 instance Run a script every hour to query the table for changes and process the documents
B. Enable a DynamoDB stream on the table Invoke an AWS Lambda function to process the documents.
C. Update the application to send a PutEvents request to Amazon EventBridge. Create an EventBridge rule to invoke an AWS Lambda function to process the documents.
D. Update the application to synchronously process the documents directly after the DynamoDB write
Answer: (SHOW ANSWER)
* DynamoDB Streams: Capture near real-time changes to DynamoDB tables, triggering downstream actions.

* Lambda for Processing: Lambda functions provide a serverless way to execute code in response to events like DynamoDB Stream updates.
* Minimal Code Changes: This solution requires the least modifications to the existing application.
References:
DynamoDB Streams:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html AWS Lambda: https://aws.amazon.com/lambda/

NEW QUESTION: 163
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 t0 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 (LEAVE A REPLY)
This solution will meet the requirements by ensuring that all traffic between users and CloudFront, and all traffic between CloudFront and the web application, are encrypted using HTTPS protocol. The Origin Protocol Policy determines how CloudFront communicates with the origin server (the web application), and setting it to "HTTPS Only" will force CloudFront to use HTTPS for every request to the origin server. The Viewer Protocol Policy determines how CloudFront responds to HTTP or HTTPS requests from users, and setting it to "HTTPS Only" or "Redirect HTTP to HTTPS" will force CloudFront to use HTTPS for every response to users. Option A is not optimal because it will use AWS KMS to encrypt traffic between CloudFront and the web application, which is not necessary or supported by CloudFront. Option C is not optimal because it will set the origin's HTTP port to 443, which is incorrect as port 443 is used for HTTPS protocol, not HTTP protocol. Option E is not optimal because it will enable the CloudFront option Restrict Viewer Access, which is used for controlling access to private content using signed URLs or signed cookies, not for encrypting traffic.
References: [Using HTTPS with CloudFront], [Restricting Access to Amazon S3 Content by Using an Origin Access Identity]

NEW QUESTION: 164

A company has developed an application that uses AWS Lambda functions to process messages from an Amazon SQS queue. One of the Lambda functions makes a call to an external API that is expected to encounter temporary service unavailability.
A developer needs to configure the function to retry failed messages from an Amazon SQS dead-letter queue.
The developer notices that the Lambda function is re-processing some messages in the queue more than once.
Which solution will resolve this issue?
A. Set a message retention period for each message. Configure the Lambda function to add a MessageId to each message.
B. Set the visibility timeout parameter at the queue level. Configure the Lambda function to delete processed messages from the queue.
C. Set a receive message wait time for each message. Configure the Lambda function to add a MessageId to each message.
D. Set the delivery delay parameter at the queue level. Configure the Lambda function to delete processed messages from the queue.
Answer: (SHOW ANSWER)
* Why Option B is Correct:Setting the visibility timeout ensures that once a message is being processed, it is temporarily hidden from other consumers. The Lambda function must delete processed messages to avoid re-processing when the visibility timeout expires.
* Why Other Options are Incorrect:
* Option A: Message retention affects how long messages stay in the queue, not how they are processed.
* Option C: Receive message wait time optimizes long polling but does not prevent re-processing.
* Option D: Delivery delay introduces latency for new messages and does not address message re- processing.
* AWS Documentation References:
* Using SQS Visibility Timeout

NEW QUESTION: 165
A developer manages a serverless application that uses an AWS Lambda function. The application periodically interacts with an external API by using short-lived authentication keys. Currently, the developer embeds the authentication keys directly in the Lambda function code. This approach requires manual updates and introduces security risks and operational inefficiencies.
The developer needs a secure and automated solution for authentication key storage, retrieval, and rotation.
Which solution will meet these requirements?

A. Store the authentication keys in AWS Secrets Manager. Configure the Lambda function to retrieve and cache the keys by using Lambda extensions.
B. Store the authentication keys in an Amazon S3 bucket. Configure the Lambda function to retrieve the keys from the bucket during each invocation.
C. Store the authentication keys in Lambda environment variables and manually update the values when needed.
D. Store the authentication keys in AWS Systems Manager Parameter Store. Configure the Lambda function to retrieve the keys during every invocation.
Answer: (SHOW ANSWER)
AWS Secrets Manager is purpose-built for securely storing and managing sensitive information such as API keys, tokens, and credentials. It provides automatic secret rotation, fine-grained IAM access control, encryption at rest using AWS KMS, and audit logging through AWS CloudTrail. This makes it the most secure and scalable option for managing short-lived authentication keys.
AWS Lambda extensions enable efficient secret retrieval by caching secrets locally within the execution environment. This reduces repeated calls to Secrets Manager on every invocation, improving performance and lowering costs while maintaining up-to-date credentials. AWS documentation explicitly recommends using Lambda extensions or caching logic for secrets that are frequently accessed.
Storing secrets in S3 (Option B) or environment variables (Option C) lacks built-in rotation and increases exposure risk. Parameter Store (Option D) supports encryption but does not provide native rotation for secrets without custom automation, making it less suitable for this use case.
Therefore, Secrets Manager with Lambda extensions provides the most secure, automated, and operationally efficient solution.

NEW QUESTION: 166
A company is building a scalable data management solution by using AWS services to improve the speed and agility of development. The solution will ingest large volumes of data from various sources and will process this data through multiple business rules and transformations.
The solution requires business rules to run in sequence and to handle reprocessing of data if errors occur when the business rules run. The company needs the solution to be scalable and to require the least possible maintenance.
Which AWS service should the company use to manage and automate the orchestration of the data flows to meet these requirements?
A. AWS Batch
B. AWS Step Functions
C. AWS Glue
D. AWS Lambda
Answer: (SHOW ANSWER)

https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html

NEW QUESTION: 167
A developer is building a serverless application that is based on AWS Lambda. The developer initializes the AWS software development kit (SDK) outside of the Lambda handcar function.
What is the PRIMARY benefit of this action?
A. Improves legibility and systolic convention
B. Takes advantage of runtime environment reuse
C. Provides better error handling
D. Creates a new SDK instance for each invocation
Answer: B (LEAVE A REPLY)
This benefit occurs when initializing the AWS SDK outside of the Lambda handler function because it allows the SDK instance to be reused across multiple invocations of the same function. This can improve performance and reduce latency by avoiding unnecessary initialization overhead. If the SDK is initialized inside the handler function, it will create a new SDK instance for each invocation, which can increase memory usage and execution time.
Reference: [AWS Lambda execution environment], [Best Practices for Working with AWS Lambda Functions]

NEW QUESTION: 168
A company has a serverless application that uses Amazon API Gateway backed by AWS Lambda proxy integration. The company is developing several backend APIs. The company needs a landing page to provide an overview of navigation to the APIs.
A developer creates a new /LandingPage resource and a new GET method that uses mock integration.
What should the developer do next to meet these requirements?
A. Configure the integration request mapping template with Content-Type of text/html and statusCode of

200. Configure the integration response mapping template with Content-Type of application/json. In the integration response mapping template, include the LandingPage HTML code that references the APIs.
B. Configure the integration request mapping template with Content-Type of text/html. In the integration request mapping template, include the LandingPage HTML code that references the APIs. Configure the integration response mapping template with Content-Type of application/json and statusCode of 200.
C. Configure the integration request mapping template with Content-Type of application/json and statusCode of 200. Configure the integration response mapping template with Content-Type of text
/html. In the integration response mapping template, include the LandingPage HTML code that references the APIs.
D. Configure the Integration request mapping template with Content-Type of application/json. In the integration request mapping template, include the LandingPage HMTL code that references the APIs.
Configure the integration response mapping template with Content-Type of text/html and statusCode of
200.
Answer: C (LEAVE A REPLY)

NEW QUESTION: 169
A developer wants to deploy a new version of an AWS Elastic Beanstalk application. During deployment the application must maintain full capacity and avoid service interruption. Additionally, the developer must minimize the cost of additional resources that support the deployment.
Which deployment method should the developer use to meet these requirements?
A. All at once
B. Rolling with additional batch
C. Bluegreen
D. Immutable
Answer: (SHOW ANSWER)
This solution will meet the requirements by using a rolling with additional batch deployment method, which deploys the new version of the application to a separate group of instances and then shifts traffic to those instances in batches. This way, the application maintains full capacity and avoids service interruption during deployment, as well as minimizes the cost of additional resources that support the deployment. Option A is not optimal because it will use an all at once deployment method, which deploys the new version of the application to all instances simultaneously, which may cause service interruption or downtime during deployment. Option C is not optimal because it will use a blue/green deployment method, which deploys the new version of the application to a separate environment and then swaps URLs with the original environment, which may incur more costs for additional

resources that support the deployment. Option D is not optimal because it will use an immutable deployment method, which deploys the new version of the application to a fresh group of instances and then redirects traffic to those instances, which may also incur more costs for additional resources that support the deployment.
References: AWS Elastic Beanstalk Deployment Policies

NEW QUESTION: 170
A developer is creating a template that uses AWS CloudFormation to deploy an application. The application is serverless and uses Amazon API Gateway, Amazon DynamoDB, and AWS Lambda.
Which AWS service or tool should the developer use to define serverless resources in YAML?
A. CloudFormation serverless intrinsic functions
B. AWS Elastic Beanstalk
C. AWS Serverless Application Model (AWS SAM)
D. AWS Cloud Development Kit (AWS CDK)
Answer: C (LEAVE A REPLY)
AWS Serverless Application Model (AWS SAM) is an open-source framework that enables developers to build and deploy serverless applications on AWS. AWS SAM uses a template specification that extends AWS CloudFormation to simplify the definition of serverless resources such as API Gateway, DynamoDB, and Lambda. The developer can use AWS SAM to define serverless resources in YAML and deploy them using the AWS SAM CLI.
References:
* [What Is the AWS Serverless Application Model (AWS SAM)? - AWS Serverless Application Model]
* [AWS SAM Template Specification - AWS Serverless Application Model]

NEW QUESTION: 171
An application that runs on AWS receives messages from an Amazon Simple Queue Service (Amazon SQS) queue and processes the messages in batches. The application sends the data to another SQS queue to be consumed by another legacy application. The legacy system can take up to 5 minutes to process some transaction data.
A developer wants to ensure that there are no out-of-order updates in the legacy system. The developer cannot alter the behavior of the legacy system.
Which solution will meet these requirements?
A. Use an SQS FIFO queue. Configure the visibility timeout value.
B. Use an SQS standard queue with a SendMessageBatchRequestEntry data type. Configure the DelaySeconds values.
C. Use an SQS standard queue with a SendMessageBatchRequestEntry data type. Configure the visibility timeout value.

D. Use an SQS FIFO queue. Configure the DelaySeconds value.

Answer: (SHOW ANSWER)

An SQS FIFO queue is a type of queue that preserves the order of messages and ensures that each message is delivered and processed only once1. This is suitable for the scenario where the developer wants to ensure that there are no out-of-order updates in the legacy system.

The visibility timeout value is the amount of time that a message is invisible in the queue after a consumer receives it2. This prevents other consumers from processing the same message simultaneously. If the consumer does not delete the message before the visibility timeout expires, the message becomes visible again and another consumer can receive it2.

In this scenario, the developer needs to configure the visibility timeout value to be longer than the maximum processing time of the legacy system, which is 5 minutes. This will ensure that the message remains invisible in the queue until the legacy system finishes processing it and deletes it. This will prevent duplicate or out-of-order processing of messages by the legacy system.

NEW QUESTION: 172

A developer is configuring an AWS Lambda function to consume messages from an Amazon SQS queue by using an event source mapping. The function consumes up to 10 messages for each invocation. The function sends data from each message to a third-party API.

The developer must configure at-least-once delivery of data to the API. The developer needs a solution that minimizes unnecessary API calls.

Which solution will meet these requirements?

A. Set up the SQS queue as a FIFO queue. Enable content-based deduplication for messages in the queue.

B. Configure reserved concurrency for the function. Set the reserved concurrency value to 1.

C. Enable reporting for batch item failures in the function event source mapping. Configure the function to report batch item failures.

D. Create a second SQS queue. Configure the original SQS queue to use the second queue as a dead-letter queue.

Answer: C (LEAVE A REPLY)

With Lambda's SQS event source mapping, messages are delivered with at-least-once semantics. When processing messages in batches (up to 10 per invocation), a common source of unnecessary downstream API calls occurs when only some records in the batch fail. Without special handling, a failure can cause the entire batch to be retried, which means records that were already successfully processed can be delivered again, leading to duplicate calls to the third-party API.

To minimize unnecessary API calls while still preserving at-least-once delivery, the best solution is to enable partial batch response (also described as "report batch item failures"). When enabled, the Lambda function returns a response that explicitly identifies which message(s) failed (by messageId). The event source mapping then retries only the failed messages, while successfully processed messages are deleted from the queue. This reduces duplicates and avoids re-sending already successful records to the third-party API, while still ensuring failed items are retried until they succeed or are moved to a DLQ (if configured).
Option A (FIFO + content-based deduplication) helps prevent duplicate enqueue events within the deduplication window, but it does not stop duplicate deliveries caused by retries, visibility timeouts, or partial failures in a batch. Option B (reserved concurrency = 1) reduces parallelism but does not prevent retries or batch-level duplication; it can also hurt throughput. Option D (DLQ) is important for poison-pill handling but does not reduce duplicate API calls during normal retry behavior; it merely captures messages after the maximum receive count.
Therefore, C is the correct choice: enable batch item failure reporting and implement the function to return partial batch responses so only failed messages are retried, minimizing unnecessary third-party API calls while maintaining at-least-once delivery.

NEW QUESTION: 173
A company has implemented AWS CodeDeploy as part of its CI/CD pipeline. The company uses automatic rollbacks during an in-place deployment of a new version of a web application on Amazon EC2 instances.
What happens if the deployment of the new version fails validation?
A. CodeDeploy restores the last successful deployment from a snapshot stored in Amazon S3.
B. CodeDeploy switches Amazon Route 53 alias records back to the previous green deployment.
C. CodeDeploy redeploys the last known stable version of the application as a new deployment with a new deployment ID.
D. AWS CodePipeline promotes the most recent SUCCEEDED deployment to production.
Answer: C (LEAVE A REPLY)
Comprehensive and Detailed Explanation (250-300 words):
For in-place deployments, AWS CodeDeploy updates instances directly rather than using blue/green infrastructure. AWS documentation states that when automatic rollback is enabled and a deployment fails (for example, due to failed health checks or validation scripts), CodeDeploy initiates a rollback deployment.
In an in-place rollback, CodeDeploy redeploys the last known successful application revision to the affected instances. This rollback is treated as a new deployment, complete with its own deployment ID and lifecycle events. CodeDeploy does not use snapshots, Route 53 switching, or external promotion logic for in- place deployments.

Options A and B describe mechanisms that are used in other services or blue/green strategies, not in-place deployments. Option D is incorrect because CodePipeline orchestrates stages but does not perform rollback logic for CodeDeploy.
Therefore, redeploying the previous stable version as a new deployment is the correct behavior.

NEW QUESTION: 174
A developer at a company needs to create a small application mat makes the same API call once each flay at a designated time. The company does not have infrastructure in the AWS Cloud yet, but the company wants to implement this functionality on AWS.
Which solution meets these requirements in the MOST operationally efficient manner?
A. Use a Kubermetes cron job that runs on Amazon Elastic Kubemetes Sen/ice (Amazon EKS)
B. Use an Amazon Linux crontab scheduled job that runs on Amazon EC2
C. Use an AWS Lambda function that is invoked by an Amazon EventBridge scheduled event.
D. Use an AWS Batch job that is submitted to an AWS Batch job queue.
Answer: (SHOW ANSWER)
This solution meets the requirements in the most operationally efficient manner because it does not require any infrastructure provisioning or management. The developer can create a Lambda function that makes the API call and configure an EventBridge rule that triggers the function once a day at a designated time. This is a serverless solution that scales automatically and only charges for the execution time of the function.
Reference: [Using AWS Lambda with Amazon EventBridge], [Schedule Expressions for Rules]

NEW QUESTION: 175
A developer is creating an AWS Serverless Application Model (AWS SAM) template. The AWS SAM template contains the definition of multiple AWS Lambda functions, an Amazon S3 bucket, and an Amazon CtoudFront distribution. One of the Lambda functions runs on Lambda@Edge in the CloudFront distribution.
The S3 bucket is configured as an origin for the CloudFront distribution.
When the developer deploys the AWS SAM template in the eu-west-1 Region, the creation of the stack fails.
Which of the following could be the reason for this issue?
A. Lambda@Edge functions can be created only in the us-east-1 Region.
B. The CloudFront distribution and the S3 bucket cannot be created in the same Region.
C. CloudFront distributions can be created only in the us-east-1 Region.
D. A single AWS SAM template cannot contain multiple Lambda functions.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 176
A company needs to distribute firmware updates to its customers around the world.
Which service will allow easy and secure control of the access to the downloads at the lowest cost?
A. Use Amazon CloudFront with signed URLs for Amazon S3.
B. Create a dedicated Amazon CloudFront Distribution for each customer.
C. Use Amazon CloudFront with AWS Lambda@Edge.
D. Use Amazon API Gateway and AWS Lambda to control access to an S3 bucket.
Answer: A (LEAVE A REPLY)
This solution allows easy and secure control of access to the downloads at the lowest cost because it uses a content delivery network (CDN) that can cache and distribute firmware updates to customers around the world, and uses a mechanism that can restrict access to specific files or versions. Amazon CloudFront is a CDN that can improve performance, availability, and security of web applications by delivering content from edge locations closer to customers. Amazon S3 is a storage service that can store firmware updates in buckets and objects. Signed URLs are URLs that include additional information, such as an expiration date and time, that give users temporary access to specific objects in S3 buckets. The developer can use CloudFront to serve firmware updates from S3 buckets and use signed URLs to control who can download them and for how long.
Creating a dedicated CloudFront distribution for each customer will incur unnecessary costs and complexity.
Using Amazon CloudFront with AWS Lambda@Edge will require additional programming overhead to implement custom logic at the edge locations. Using Amazon API Gateway and AWS Lambda to control access to an S3 bucket will also require additional programming overhead and may not provide optimal performance or availability.
Reference: [Serving Private Content through CloudFront], [Using CloudFront with Amazon S3]

NEW QUESTION: 177
A developer is writing an application for a company. The application will be deployed on Amazon EC2 and will use an Amazon RDS for Microsoft SQL Server database. The company's security team requires that database credentials are rotated at least weekly.
How should the developer configure the database credentials for this application?
A. Create a database user. Store the username and password in an AWS Systems Manager Parameter Store secure string parameter. Enable rotation of the AWS KMS key that is used to encrypt the parameter.
B. Enable IAM authentication for the database. Create a database user for use with IAM authentication.
Enable password rotation.
C. Create a database user. Store the username and password in an AWS Secrets Manager secret that has daily rotation enabled.

D. Use the EC2 user data to create a database user. Provide the username and password in environment variables to the application.

Answer: C (LEAVE A REPLY)

The requirement is to rotate database credentials at least weekly. The AWS service that is purpose-built for storing and automatically rotating secrets is AWS Secrets Manager. Secrets Manager supports rotation schedules (including daily, weekly, or custom intervals) and integrates with supported databases through rotation Lambda functions. With rotation enabled, Secrets Manager can periodically generate a new password, update the secret value, and apply the new credential to the database user, while allowing applications to retrieve the current credential securely at runtime.

Option C matches this best practice: create a database user, store the username/password in a Secrets Manager secret, and enable rotation (daily rotation more than satisfies a weekly requirement). Applications on EC2 can retrieve the secret at startup or on a refresh interval using the AWS SDK, and can cache it according to best practices to reduce API calls, while still supporting rotation.

Option A (Parameter Store secure string) encrypts parameters, but Parameter Store does not provide the same managed secret rotation workflow for database credentials; rotating the KMS key is not the same as rotating the database password. Option B is not applicable as stated for RDS for Microsoft SQL Server: IAM database authentication is supported for specific engines (commonly MySQL and PostgreSQL) and is not the standard mechanism for SQL Server password rotation. Option D is insecure and operationally fragile: embedding long-lived credentials in user data and environment variables makes rotation difficult and increases exposure risk.

Therefore, the most secure and compliant configuration is C: store DB credentials in AWS Secrets Manager and enable automated rotation on an interval that meets or exceeds the weekly rotation requirement.

NEW QUESTION: 178

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 (LEAVE A REPLY)

The requirements are: (1) store an API key that a Lambda function will use, (2) ensure the secret is encrypted at rest using AWS KMS, and (3) the company must control key rotation, which implies using a customer managed KMS key (CMK) rather than an AWS managed key.

Option C meets the requirements by storing the API key in AWS Systems Manager Parameter Store as a SecureString parameter encrypted with a customer managed KMS key. SecureString is designed for sensitive configuration data and integrates with KMS so the organization can choose the CMK, manage its lifecycle, and control rotation policies. The Lambda function can retrieve the parameter at runtime using the AWS SDK, and IAM policies can tightly control access to the parameter and to the KMS key.

Option E also meets the requirements by storing the API key in a Lambda environment variable encrypted with a customer managed KMS key. Lambda encrypts environment variables at rest and allows you to specify a customer managed KMS key for encryption. This gives the company control over key rotation and key policy, satisfying the "must control key rotation" requirement. The function reads the value from the environment at runtime without additional network calls.

Why the other options fail:

* A uses an AWS managed KMS key, which does not satisfy the requirement for the company to control rotation (you cannot manage rotation of AWS managed keys in the same way).
* B is a plain String parameter, which is not encrypted as a secret and does not meet the at-rest encryption requirement.
* D uses an AWS managed KMS key, again failing the company-controlled rotation requirement.

Therefore, the two valid solutions are C (Parameter Store SecureString with a customer managed CMK) and E (Lambda environment variable encrypted with a customer managed CMK).

NEW QUESTION: 179

A developer has created an AWS Lambda function to provide notification through Amazon Simple Notification Service (Amazon SNS) whenever a file is uploaded to Amazon S3 that is larger than 50 MB. The developer has deployed and tested the Lambda function by using the CLI. However, when the event notification is added to the S3 bucket and a 3.000 MB file is uploaded, the Lambda function does not launch.

Which of the following Is a possible reason for the Lambda function's inability to launch?

A. The S3 event notification does not activate for files that are larger than 1.000 MB.

B. The S3 bucket needs to be made public.

C. Lambda functions cannot be invoked directly from an S3 event.

D. The resource-based policy for the Lambda function does not have the required permissions to be invoked by Amazon S3.

Answer: (SHOW ANSWER)

NEW QUESTION: 180
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 (LEAVE A REPLY)
Step-by-Step Breakdown:
Requirement Summary:
In-transit encryption (when data is moving between client and S3)
Encryption at rest using AWS KMS keys (which are rotatable on-demand)
Option A: Write an S3 bucket policy to allow only encrypted connections over HTTPS by using permissions boundary Incorrect: Permissions boundaries are for IAM policy control, not S3 bucket policies.
Also, encryption enforcement in-transit is not achieved through permissions boundaries.
Option B: Configure an S3 bucket policy to enable client-side encryption Incorrect: AWS does not enforce client-side encryption via S3 bucket policies.
Client-side encryption must be handled entirely by the application.
Also, KMS is for server-side encryption, not client-side.
Option C: Configure the application to encrypt the objects by using an AWS KMS customer managed key before uploading the objects Correct: This fulfills the requirement of encrypting objects at rest using a KMS CMK, which can be rotated.
You can specify the KMS key using the SSE-KMS option when putting an object.
Option D: Write an S3 bucket policy to allow only encrypted connections over HTTPS by using the aws:
SecureTransport condition
Correct: This enforces in-transit encryption, which ensures that only HTTPS connections are allowed.

This is the AWS-recommended way to enforce encryption in transit via bucket policy.
Option E: Configure S3 Block Public Access settings for the S3 bucket to allow only encrypted connections over HTTPS Incorrect: Block Public Access is used to prevent public exposure of the bucket.
It has nothing to do with encryption enforcement, whether in transit or at rest.
Using SSE-KMS:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html
aws:SecureTransport condition in bucket policies:
https://docs.aws.amazon.com/AmazonS3/latest/userguide
/amazon-s3-policy-keys.html
Server-side encryption with AWS KMS:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side- encryption.html

NEW QUESTION: 181
A healthcare company is developing a multi-tier web application to manage patient records that are in an Amazon Aurora PostgreSQL database cluster. The company stores the application code in a Git repository and deploys the code to Amazon EC2 instances.
The application must comply with security policies and follow the principle of least privilege. The company must securely manage database credentials and API keys within the application code. The company must have the ability to rotate encryption keys on demand.
Which solution will meet these requirements?
A. Store database credentials and API keys in AWS Secrets Manager. Use AWS managed AWS KMS keys. Set up automatic key rotation. Use the AWS SDK to retrieve secrets.
B. Store the database credentials and API keys in AWS Secrets Manager. Use customer managed AWS KMS keys. Set up automatic key rotation. Create a key policy in the application to retrieve secrets by using the AWS SDK.
C. Store the database credentials in the application code. Separate credentials by using environment- specific branches that have restricted access to the code repositories.
D. Store the database credentials and API keys as parameters in AWS Systems Manager Parameter Store.Encrypt the credentials and API keys with AWS managed AWS KMS keys. Use the AWS SDK to retrieve secrets.
Answer: A (LEAVE A REPLY)
Requirement Summary:
Multi-tier app on EC2 + Aurora PostgreSQL
Must comply with least privilege and security policies
Need to manage credentials and API keys securely
Must support key rotation on demand
Evaluate Options:
A). Secrets Manager + AWS managed KMS keys
Best practice for secure secret storage

Supports auto rotation
Uses AWS SDK to fetch at runtime (secure, avoids hardcoding)
AWS managed keys are rotated automatically and easier to manage
B). Secrets Manager + customer managed keys
Also valid, but adds complexity
Since the question asks for LEAST development effort, AWS-managed keys are preferred C). Store secrets in code Violates all security best practices D). Use SSM Parameter Store + AWS managed keys Possible, but Secrets Manager is preferred when rotation is needed Parameter Store does not natively rotate secrets Secrets Manager: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html Automatic key rotation: https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html Best practices for secret management: https://docs.aws.amazon.com/secretsmanager/latest/userguide/best- practices.html

NEW QUESTION: 182
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 sialic content in Amazon S3
Answer: D,E (LEAVE A REPLY)
The combination of steps that will resolve the latency issue is to create an Amazon CloudFront distribution to cache the static content and store the application's static content in Amazon S3. This way, the company can use CloudFront to deliver the static content from edge locations that are closer to the website users, reducing latency and improving performance. The company can also use S3 to store the static content reliably and cost-effectively, and integrate it with CloudFront easily. The other options either do not address the latency issue, or are not necessary or feasible for the given scenario.

Reference: Using Amazon S3 Origins and Custom Origins for Web Distributions

NEW QUESTION: 183
A bookstore has an ecommerce website that stores order information in an Amazon DynamoDB table named BookOrders. The DynamoDB table contains approximately one million records.
The table uses OrderID as a partition key. There are no other indexes.
A developer wants to build a new reporting feature to retrieve all records from the table for a specified customer, based on a CustomerID property.
A. Create a DynamoDB global secondary index (GSI) on the table. Use CustomerID as the partition key.
Use the specified CustomerID value to run a query on the table.
B. Create a DynamoDB global secondary index (GSI) on the table. Use CustomerID as the sort key. Use a filter expression to perform a scan operation on the table to match on the specified CustomerID value.
C. Create a DynamoDB local secondary index (LSI) on the table. Use CustomerID as the sort key. Run a PartiQL query on the table with a SELECT statement where CustomerID equals the specified CustomerID value.
D. Create a DynamoDB local secondary index (LSI) on the table. Use CustomerID as the partition key.Use the specified CustomerID value to run a query on the table.
Answer: A (LEAVE A REPLY)
Comprehensive and Detailed Step-by-Step Explanation:
The requirement is to query records by CustomerID, which is not the current partition key (OrderID). To achieve this efficiently:
* Option A: Create a GSI with CustomerID as the Partition Key:
* A Global Secondary Index (GSI) allows developers to create a different partition key and optional sort key for querying the data.
* By creating a GSI with CustomerID as the partition key, the developer can query the table efficiently using CustomerID as the primary lookup key.
* This avoids scanning the entire table and matches the requirement.
* Why Other Options Are Incorrect:
* Option B: Using CustomerID as a sort key for the GSI and performing a scan operation is inefficient. Queries are optimized, but scans are not.
* Option C and D: Local Secondary Indexes (LSI) are only valid when the partition key remains the same as the base table. Since OrderID is the base table's partition key, using CustomerID as the partition key or sort key in an LSI is not valid.
References:
Amazon DynamoDB Documentation: GSIs

NEW QUESTION: 184

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 UI 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 UI team reports that the request to process a file is often returning timeout errors because of the see or complexity of the files. The UI team wants the API to provide an immediate response so that the UI 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 (LEAVE A REPLY)
This solution allows the API to invoke the Lambda function asynchronously, which means that the API will return an immediate response without waiting for the function to complete. The X-Amz-Invocation-Type header specifies the invocation type of the Lambda function, and setting it to 'Event' means that the function will be invoked asynchronously. The function can then use Amazon Simple Email Service (SES) to send an email message when the report processing is complete.
Reference: [Asynchronous invocation], [Set up Lambda proxy integrations in API Gateway]

NEW QUESTION: 185
A developer is planning to migrate on-premises company data to Amazon S3. The data must be encrypted, and the encryption Keys must support automate annual rotation. The company must use AWS Key Management Service (AWS KMS) to encrypt the data.
When type of keys should the developer use to meet these requirements?
A. Amazon S3 managed keys
B. Symmetric customer managed keys with key material that is generated by AWS
C. Asymmetric customer managed keys with key material that generated by AWS
D. Symmetric customer managed keys with imported key material

Answer: (SHOW ANSWER)
The type of keys that the developer should use to meet the requirements is symmetric customer managed keys with key material that is generated by AWS. This way, the developer can use AWS Key Management Service (AWS KMS) to encrypt the data with a symmetric key that is managed by the developer. The developer can also enable automatic annual rotation for the key, which creates new key material for the key every year. The other options either involve using Amazon S3 managed keys, which do not support automatic annual rotation, or using asymmetric keys or imported key material, which are not supported by S3 encryption.
Reference: Using AWS KMS keys to encrypt S3 objects

NEW QUESTION: 186
A developer needs to give a new application the ability to retrieve configuration data. The application must be able to retrieve new configuration data values without the need to redeploy the application code. If the application becomes unhealthy because of a bad configuration change, the developer must be able to automatically revert the configuration change to the previous value.
A. Use AWS Secrets Manager to manage and store the configuration data. Integrate Secrets Manager with a custom AWS Config rule that has remediation actions to track changes in the application and to roll back any bad configuration changes.
B. Use AWS Secrets Manager to manage and store the configuration data. Integrate Secrets Manager with a custom AWS Config rule. Attach a custom AWS Systems Manager document to the rule that automatically rolls back any bad configuration changes.
C. Use AWS AppConfig to manage and store the configuration data. Integrate AWS AppConfig with Amazon CloudWatch to monitor changes to the application. Set up an alarm to automatically roll back any bad configuration changes.
D. Use AWS AppConfig to manage and store the configuration data. Integrate AWS AppConfig with Amazon CloudWatch to monitor changes to the application. Set up CloudWatch Application Signals to roll back any bad configuration changes.
Answer: D (LEAVE A REPLY)
Comprehensive and Detailed Step-by-Step Explanation:
* Option D: AWS AppConfig with CloudWatch Application Signals
* AWS AppConfig is designed for managing and deploying application configurations dynamically, without redeployment.
* CloudWatch Application Signals provide automatic rollback mechanisms in case of an unhealthy application state due to bad configuration changes.
* This solution meets the requirements with minimal operational overhead by ensuring both dynamic updates and rollback functionality.
* Why Other Options Are Incorrect:
* Option A and B: AWS Secrets Manager is designed for secrets management, not dynamic application configuration. Custom Config rules add unnecessary complexity.

* Option C: While CloudWatch alarms can monitor application changes, using alarms for rollback requires manual setup and lacks the automatic rollback provided by Application Signals.
References:
* AWS AppConfig Documentation

NEW QUESTION: 187
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 (LEAVE A REPLY)
Increasing the number of shards of the Kinesis data stream will increase the throughput and parallelism of the data processing. Increasing the memory that is allocated to the Lambda function will also increase the CPU and network performance of the function, which will reduce the run duration and improve the processing speed. Option B is not correct because decreasing the timeout of the Lambda function will not affect the processing speed, but may cause some records to fail if they exceed the timeout limit. Option D is not correct because decreasing the number of shards of the Kinesis data stream will decrease the throughput and parallelism of the data processing, which will slow down the processing speed. Option E is not correct because increasing the timeout of the Lambda function will not affect the processing speed, but may increase the cost of running the function.
References: [Amazon Kinesis Data Streams Scaling], [AWS Lambda Performance Tuning]

NEW QUESTION: 188
A developer is preparing to begin development of a new version of an application. The previous version of the application is deployed in a production environment. The developer needs to deploy fixes and updates to the current version during the development of the new version of the application. The code for the new version of the application is stored in AWS CodeCommit.
Which solution will meet these requirements?
A. From the main branch, create a feature branch for production bug fixes. Create a second feature branch from the main branch for development of the new version.

B. Create a Git tag of the code that is currently deployed in production. Create a Git tag for the development of the new version. Push the two tags to the CodeCommit repository.
C. From the main branch, create a branch of the code that is currently deployed in production. Apply an IAM policy that ensures no other other users can push or merge to the branch.
D. Create a new CodeCommit repository for development of the new version of the application. Create a Git tag for the development of the new version.
Answer: A (LEAVE A REPLY)
A feature branch is a branch that is created from the main branch to work on a specific feature or task1. Feature branches allow developers to isolate their work from the main branch and avoid conflicts with other changes1. Feature branches can be merged back to the main branch when the feature or task is completed and tested1.
In this scenario, the developer needs to maintain two parallel streams of work: one for fixing and updating the current version of the application that is deployed in production, and another for developing the new version of the application. The developer can use feature branches to achieve this goal.
The developer can create a feature branch from the main branch for production bug fixes. This branch will contain the code that is currently deployed in production, and any fixes or updates that need to be applied to it. The developer can push this branch to the CodeCommit repository and use it to deploy changes to the production environment.
The developer can also create a second feature branch from the main branch for development of the new version of the application. This branch will contain the code that is under development for the new version, and any changes or enhancements that are part of it. The developer can push this branch to the CodeCommit repository and use it to test and deploy the new version of the application in a separate environment.
By using feature branches, the developer can keep the main branch stable and clean, and avoid mixing code from different versions of the application. The developer can also easily switch between branches and merge them when needed.

NEW QUESTION: 189
A developer maintains an Amazon API Gateway REST API. Customers use the API through a frontend UI and Amazon Cognito authentication.
The developer has a new version of the API that contains new endpoints and backward-incompatible interface changes. The developer needs to provide beta access to other developers on the team without affecting customers.
Which solution will meet these requirements with the LEAST operational overhead?
A. Define a development stage on the API Gateway API. Instruct the other developers to point the endpoints to the development stage.
B. Define a new API Gateway API that points to the new API application code. Instruct the other developers to point the endpoints to the new API.

C. Implement a query parameter in the API application code that determines which code version to call.
D. Specify new API Gateway endpoints for the API endpoints that the developer wants to add.
Answer: (SHOW ANSWER)
Amazon API Gateway is a service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. The developer can define a development stage on the API Gateway API and instruct the other developers to point the endpoints to the development stage. This way, the developer can provide beta access to the new version of the API without affecting customers who use the production stage. This solution will meet the requirements with the least operational overhead.
References:
* [What Is Amazon API Gateway? - Amazon API Gateway]
* [Set up a Stage in API Gateway - Amazon API Gateway]

NEW QUESTION: 190
A developer wants to reduce risk when deploying a new version of an existing AWS Lambda function. To test the Lambda function, the developer needs to split the traffic between the existing version and the new version of the Lambda function.
Which solution will meet these requirements?
A. Configure a weighted routing policy in Amazon Route 53. Associate the versions of the Lambda function with the weighted routing policy.
B. Create a function alias. Configure the alias to split the traffic between the two versions of the Lambda function.
C. Create an Application Load Balancer (ALB) that uses the Lambda function as a target. Configure the ALB to split the traffic between the two versions of the Lambda function.
D. Create the new version of the Lambda function as a Lambda layer on the existing version. Configure the function to split the traffic between the two layers.
Answer: (SHOW ANSWER)
AWS Lambda supports versions and aliases to manage deployments safely. An alias is a named pointer to a specific function version (for example, prod # version 5). Importantly, Lambda aliases can be configured with routing configuration to split incoming invocation traffic between two versions. This enables canary or linear deployments where most traffic goes to the stable version and a small percentage goes to the new version for testing. If issues occur, the developer can quickly shift traffic back by updating the alias.
Option B is the native, simplest, and recommended solution for Lambda traffic shifting.
Option A is not correct because Route 53 weighted routing can split DNS queries between endpoints, but it does not natively target Lambda versions directly for most invocation patterns, and it's not the standard mechanism for Lambda version traffic shifting.

Option C is unnecessary and adds operational overhead. ALB can invoke Lambda targets, but splitting between Lambda versions is more naturally handled by aliases without provisioning and managing an ALB.
Option D misunderstands layers: layers are for dependencies/shared code, not for deploying new versions or traffic splitting.
Therefore, using a Lambda alias with weighted routing between versions meets the requirement.

NEW QUESTION: 191
A company has a serverless application that uses Amazon API Gateway and AWS Lambda functions to expose a RESTful API. The company uses a continuous integration and continuous delivery (CI/CD) workflow to deploy the application to multiple environments. The company wants to implement automated integration tests after deployment.
A developer needs to set up the necessary infrastructure and processes to automate the deployment and integration tests for the serverless application.
A. Configure API Gateway stages to represent each application environment. Use AWS SAM templates to manage the infrastructure for the Lambda functions and API resources. Use AWS CodeBuild to implement automated deployment tests to validate the deployments in each stage.
B. Configure API Gateway stages to represent each application environment. Use AWS CloudFormation to manage the infrastructure for the Lambda functions and API resources. Use AWS CodeBuild to implement automated deployment tests to validate the deployments in each stage.
C. Use AWS CodePipeline to create a CI/CD pipeline. Configure API Gateway stages to represent each application environment. Use AWS CloudFormation templates to manage the infrastructure for the Lambda functions and API resources. Use AWS CodeBuild to implement automated deployment tests to validate the deployments in each stage.
D. Use AWS CloudFormation to create and deploy the application infrastructure in each application environment. Use the AWS CLI to invoke Lambda functions to perform deployment tests after each deployment.
Answer: C (LEAVE A REPLY)
Comprehensive and Detailed Step-by-Step Explanation:
Option C: Use AWS CodePipeline for CI/CD Workflow:
AWS CodePipeline automates the entire CI/CD pipeline, including build, deploy, and test stages. This minimizes manual effort and integrates well with AWS services.
API Gateway Stages: Represent different environments, such as dev, test, and prod, allowing isolated deployment and testing.
AWS CloudFormation Templates: Ensure that the infrastructure for Lambda and API Gateway is consistent across environments.

AWS CodeBuild for Automated Tests: Validates the deployments in each stage, ensuring integration and functionality are tested post-deployment.
Why Other Options Are Incorrect:
Option A and B: While using AWS SAM or CloudFormation for infrastructure management is valid, these options lack the fully automated CI/CD pipeline provided by CodePipeline.
Option D: Manually invoking Lambda functions using the AWS CLI introduces operational overhead and lacks the automation provided by CodePipeline.
References:
AWS CodePipeline Documentation
API Gateway Stages for CI/CD

NEW QUESTION: 192
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. Move the user credentials from Lambda environment variables to AWS Systems Manager Parameter Store.
B. Move the user credentials from Lambda environment variables to AWS Key Management Service (AWS KMS).
C. Remove the user credentials from the Lambda environment. Implement 1AM database authentication.
D. Move the user credentials from the Lambda environment to an encrypted .txt file. Store the file in an S3 bucket.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 193
A developer is integrating Amazon ElastiCache in an application. The cache will store data from a database.
The cached data must populate real-time dashboards. Which caching strategy will meet these requirements?
A. A write-through cache
B. A read-through cache
C. A write-behind cache
D. A lazy-loading cache
Answer: (SHOW ANSWER)

NEW QUESTION: 194

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: (SHOW ANSWER)

To meet the requirement that developers cannot view credit card numbers while the fraud detection account can, the solution must (1) ensure the sensitive data is masked at

ingestion/storage in CloudWatch Logs, and (2) allow only an authorized principal to view unmasked values.
First, the developers must attach the data protection policy to the CloudWatch log group that receives the API Gateway HTTP API access logs (and/or the Lambda log group if credit card numbers can appear there).
A CloudWatch Logs data protection policy is applied at the log group level and instructs CloudWatch Logs to identify sensitive data using managed data identifiers (for example, CreditCardNumber) and then apply the configured de-identification action (here, masking). Without attaching the policy to the log group, the masking/redaction behavior will not be enforced for newly ingested log events, and developers could still see raw request parameters.
Second, to permit only the fraud detection account to reveal the masked values when necessary, the IAM role that the fraud detection account assumes must have the specific CloudWatch Logs permission to unmask protected data. Granting logs:Unmask to that role enables authorized access to view the sensitive values while keeping them masked for everyone else who lacks that permission (such as developer accounts assuming other roles).
Options A and E are not required for CloudWatch Logs data protection enforcement; the core controls are log- group policy attachment (to perform masking) and IAM authorization (to permit unmasking only for the fraud role). Therefore, the correct combination is C (apply the policy to the log group that captures the HTTP API logs) and B (grant logs:Unmask to the fraud detection role).

NEW QUESTION: 195
A developer wrote an application that uses an AWS Lambda function to asynchronously generate short videos based on requests from customers. This video generation can take up to 10 minutes. After the video is generated, a URL to download the video is pushed to the customer's web browser. The customer should be able to access these videos for at least 3 hours after generation.
Which solution will meet these requirements?
A. Store the video in the /tmp folder within the Lambda execution environment. Push a Lambda function URL to the customer.
B. Store the video in an Amazon EFS file system attached to the function. Generate a presigned URL for the video object and push the URL to the customer.
C. Store the video in Amazon S3. Generate a presigned URL for the video object and push the URL to the customer.
D. Store the video in an Amazon CloudFront distribution. Generate a presigned URL for the video object and push the URL to the customer.
Answer: C (LEAVE A REPLY)

The customer must be able to download the generated video for at least 3 hours, which requires durable storage that persists beyond the Lambda execution environment. The best fit is Amazon S3 with a presigned URL.
Lambda's /tmp storage (Option A) is ephemeral and scoped to the execution environment. It is not intended for durable customer downloads and can be deleted when the execution environment is recycled. A Lambda function URL also does not solve durable file hosting and would require the function to serve the content itself.
Option C uses S3 to store the video object durably and then issues a presigned URL that grants time-limited access to that specific object. Presigned URLs are designed for exactly this scenario: allow unauthenticated users temporary access to a private S3 object without making the bucket public. The developer can set the URL expiration to 3 hours (or longer), meeting the access requirement cleanly.
Option B is incorrect because S3 presigned URLs are for S3 objects, not EFS files. EFS does not natively use S3-style presigned URLs for external downloads.
Option D is not the right model: CloudFront can distribute S3 content and supports signed URLs/cookies, but CloudFront is a distribution layer, not an origin storage solution by itself. You would still need to store the video in S3 (or another origin). For the stated requirement, S3 + presigned URL is simpler and has less overhead.
Therefore, store videos in Amazon S3 and provide presigned URLs with a 3-hour expiration.

NEW QUESTION: 196
A developer accesses AWS CodeCommit over SSH. The SSH keys configured to access AWS CodeCommit are tied to a user with the following permissions:
The developer needs to create/delete branches
Which specific IAM permissions need to be added based on the principle of least privilege?
A. Option A
B. Option B
C. Option C
D. Option D
Answer: A (LEAVE A REPLY)
This solution allows the developer to create and delete branches in AWS CodeCommit by granting the codecommit:CreateBranch and codecommit:DeleteBranch permissions.
These are the minimum permissions required for this task, following the principle of least privilege. Option B grants too many permissions, such as codecommit:Put*, which allows the developer to create, update, or delete any resource in CodeCommit.
Option C grants too few permissions, such as codecommit:Update*, which does not allow the developer to create or delete branches. Option D grants all permissions, such as codecommit:*, which is not secure or recommended.
Reference: [AWS CodeCommit Permissions Reference], [Create a Branch (AWS CLI)]

NEW QUESTION: 197
A developer is managing an application that uploads user files to an Amazon S3 bucket named companybucket. The company wants to maintain copies of all the files uploaded by users for compliance purposes, while ensuring users still have access to the data through the application.
Which IAM permissions should be applied to users to ensure they can create but not remove files from the bucket?
A. {"Version": "2012-10-17","Statement": [{"Sid": "statement1","Effect": "Allow","Action": ["s3:
GetObject", "s3:PutObject", "s3:DeleteObject"],"Resource": ["arn:aws:s3:::companybucket"]}]}
B. {"Version": "2012-10-17","Statement": [{"Sid": "statement1","Effect": "Allow","Action": ["s3:
CreateBucket", "s3:GetBucketLocation"],"Resource": "arn:aws:s3:::companybucket"}]}
C. {"Version": "2012-10-17","Statement": [{"Sid": "statement1","Effect": "Allow","Action": ["s3:
GetObject", "s3:PutObject", "s3:DeleteObject", "s3:PutObjectRetention"],"Resource": "arn:aws:s3:::
companybucket"}]}
D. {"Version": "2012-10-17","Statement": [{"Sid": "statement1","Effect": "Allow","Action": ["s3:
GetObject", "s3:PutObject"],"Resource": ["arn:aws:s3:::companybucket"]}]}
Answer: D (LEAVE A REPLY)
To meet the requirement:
Users must be able to upload (PutObject) and read (GetObject) files but not delete them.
Option D ensures users cannot delete files by omitting the s3:DeleteObject action while allowing s3:
GetObject and s3:PutObject.
Option A: Includes s3:DeleteObject, which allows users to delete files and does not meet the requirement.
Option B: Contains unrelated actions like CreateBucket, which is not relevant here.

Option C: Adds s3:PutObjectRetention, which is unnecessary and does not restrict DeleteObject.
Reference:AWS S3 Permissions Documentation

NEW QUESTION: 198
A developer is creating AWS CloudFormation templates to manage an application's deployment in Amazon Elastic Container Service (Amazon ECS) through AWS CodeDeploy. The developer wants to automatically deploy new versions of the application to a percentage of users before the new version becomes available for all users.
How should the developer manage the deployment of the new version?
A. Modify the CloudFormation template to include a Transform section and the AWS::CodeDeploy::
BlueGreen hook.
B. Run CloudFormation stack updates on the application stack to deploy new application versions when they are available.
C. Deploy the new version in a new CloudFormation stack. After testing is complete, update the application's DNS records for the new stack.
D. Create a nested stack for the new version. Include a Transform section and the AWS::CodeDeploy::BlueGreen hook.
Answer: (SHOW ANSWER)

NEW QUESTION: 199
A company is building a serverless application composed of multiple AWS Lambda functions. The company wants to accelerate feature development without adding staff or reducing quality. The solution must improve unit tests and code reviews, integrate with the existing IDE, and require no new infrastructure.
Which solution will meet these requirements?
A. Use AWS CodeBuild with custom scripts for unit tests and Amazon CodeGuru Reviewer for code reviews.
B. Use Amazon CodeGuru for code reviews and a custom Lambda-based solution for unit tests.
C. Use AWS CodeBuild for tests and rely on manual pull request reviews.
D. Use Amazon Q Developer Pro to generate unit tests and perform code reviews directly in the IDE.
Answer: D (LEAVE A REPLY)
Amazon Q Developer Pro is an AWS-native, generative AI-powered development assistant designed to help developers write, test, and review code directly within supported IDEs such as Visual Studio Code and JetBrains. AWS documentation highlights Amazon Q Developer Pro as a productivity tool that accelerates development without requiring additional infrastructure.

Amazon Q Developer Pro can automatically generate unit tests, suggest improvements, identify potential bugs, and provide context-aware code reviews. Because it integrates directly into the IDE, developers receive immediate feedback during development, which shortens feedback loops and improves overall code quality.
Options A, B, and C require additional tooling, scripting, or manual processes. While CodeBuild and CodeGuru Reviewer are valuable services, they introduce pipeline complexity and do not provide real-time, IDE-integrated assistance for writing tests and reviewing code.
Amazon Q Developer Pro uniquely satisfies all stated requirements:
* No infrastructure to manage
* Native IDE integration
* Automated unit test generation
* AI-assisted code reviews
Therefore, using Amazon Q Developer Pro is the most efficient and modern AWS-recommended solution.

NEW QUESTION: 200
An application that runs on AWS Lambda requires access to specific highly confidential objects in an Amazon S3 bucket. In accordance with the principle of least privilege a company grants access to the S3 bucket by using only temporary credentials.
How can a developer configure access to the S3 bucket in the MOST secure way?
A. Hardcode the credentials that are required to access the S3 objects in the application code. Use the credentials to access me required S3 objects.
B. Create a secret access key and access key ID with permission to access the S3 bucket. Store the key and key ID in AWS Secrets Manager. Configure the application to retrieve the Secrets Manager secret and use the credentials to access me S3 objects.
C. Create a Lambda function execution role Attach a policy to the rote that grants access to specific objects in the S3 bucket.
D. Create a secret access key and access key ID with permission to access the S3 bucket Store the key and key ID as environment variables m Lambda. Use the environment variables to access the required S3 objects.
Answer: (SHOW ANSWER)
This solution will meet the requirements by creating a Lambda function execution role, which is an IAM role that grants permissions to a Lambda function to access AWS resources such as Amazon S3 objects. The developer can attach a policy to the role that grants access to specific objects in the S3 bucket that are required by the application, following the principle of least privilege. Option A is not optimal because it will hardcode the credentials that are required to access S3 objects in the application code, which is insecure and difficult to maintain. Option B is not optimal because it will create a secret access key and access key ID with permission to access the S3 bucket, which will introduce additional security risks and complexity for storing and managing credentials.

Option D is not optimal because it will store the secret access key and access key ID as environment variables in Lambda, which is also insecure and difficult to maintain.
References: [AWS Lambda Execution Role], [Using AWS Lambda with Amazon S3]

NEW QUESTION: 201
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. 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.
B. Create an AWS Support ticket to increase the maximum package size.
C. 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.
D. Upload the package to Amazon S3. Use the Functions page on the Lambda console to upload the package from the S3 location.
E. Use the update-function-code AWS CLI command. Pass the -publish parameter.
Answer: C,D (LEAVE A REPLY)

NEW QUESTION: 202
An application uses an Amazon DynamoDB table to manage user profiles. A UserID attribute is the primary key of the table. The table also includes columns named Username, EmailAddress, RegistrationDate, Location, and Status.
The application needs to display a list of users from a specific location who registered after a specific date.
Queries on the table must be optimized for efficiency.
Which solution will meet these requirements?
A. Create a global secondary index (GSI). Use Location as the partition key and RegistrationDate as the sort key. Use the Query operation to retrieve the specified users.
B. Use the Scan operation to retrieve the specified users. Use a filter expression for a value in the RegistrationDate column that is greater than the date required by the application.
C. Create a local secondary index (LSI). Use Location as the partition key and RegistrationDate as the sort key. Use the Query operation to retrieve the specified users.
D. Use the BatchGetItem operation with a filter on the RegistrationDate column for a value that is greater than the required date to retrieve the specified users.
Answer: (SHOW ANSWER)

The base table's primary key is UserID, which means efficient Query operations on the table natively require specifying UserID (partition key equality). The required access pattern, however, is to retrieve many users for a given Location and then filter/sort by RegistrationDate (users who registered after a given date). Because Location is not part of the table's primary key, the efficient DynamoDB approach is to create an index that supports this access pattern.
A global secondary index (GSI) can have a different partition key and sort key than the base table. By creating a GSI with Location as the partition key and RegistrationDate as the sort key, the application can use the Query operation to fetch all users in a specific location (Location = :loc) and apply a sort key condition such as RegistrationDate > :date (or BETWEEN), which is efficient and avoids a full-table scan. This design is the canonical pattern for optimizing DynamoDB queries: model the table and indexes around known access patterns so the application can use Query instead of Scan.
Option C (LSI) is not valid for this requirement because an LSI must use the same partition key as the base table (UserID). Since the access pattern is by Location, an LSI cannot make Location the partition key.
Option B (Scan + filter) is inefficient at scale because Scan reads every item (or every item in the scanned segments) and then filters results, consuming read capacity and increasing latency. Option D (BatchGetItem) requires the caller to already know the primary keys of the items to retrieve; it cannot discover "all users in a location after a date" and does not support filtering in the way Query on an index does.
Therefore, A is the correct and most efficient solution: a GSI on (Location, RegistrationDate) and Query with a date range condition.

NEW QUESTION: 203
A developer is designing a full-stack serverless application. Files for the website are stored in an Amazon S3 bucket. AWS Lambda functions that use Amazon API Gateway endpoints return results from an Amazon DynamoDB table.
The developer must create a solution that securely provides registration and authentication for the application while minimizing the amount of configuration.
Which solution meets these requirements?
A. Create an Amazon Cognito user pool and an app client. Configure the app client to use the user pool and provide the hosted web UI provided for sign-up and sign-in.
B. Configure an Amazon Cognito identity pool. Map the users with IAM roles that are configured to access the S3 bucket that stores the website.
C. Configure and launch an Amazon EC2 instance to set up an identity provider with an Amazon Cognito user pool. Configure the user pool to provide the hosted web UI for sign-up and sign-in.
D. Create an IAM policy that allows access to the website that is stored in the S3 bucket. Attach the policy to an IAM group. Add IAM users to the group.
Answer: (SHOW ANSWER)

For a serverless web application that needs user registration and authentication with minimal configuration, the standard AWS solution is Amazon Cognito User Pools. User Pools provide managed user directories, sign- up/sign-in flows, MFA options, password reset, and token-based authentication (OAuth2/OIDC/JWT). This integrates naturally with API Gateway (Cognito authorizers) and Lambda.
Option A is best because it uses Cognito User Pools plus an app client and the Hosted UI, which provides ready-made sign-up/sign-in pages. Hosted UI minimizes custom UI and backend authentication code while still allowing branding customization and secure token issuance. The SPA can authenticate via Cognito, receive JWT tokens, and call API Gateway endpoints securely.
Option B (identity pool) is primarily for granting temporary AWS credentials to access AWS services directly (federation). It does not replace User Pools for user registration/login flows; typically identity pools are used in addition to user pools, not instead, and require more configuration.
Option C adds unnecessary infrastructure (EC2) and defeats the "minimize configuration" goal.
Option D is not appropriate for end-user authentication. IAM users/groups are for workforce/admin access, not for customer-facing apps.
Therefore, Cognito User Pool + Hosted UI is the minimal-configuration secure solution.

NEW QUESTION: 204
A developer has a legacy application that is hosted on-premises. Other applications hosted on AWS depend on the on-premises application for proper functioning. In case of any application errors, the developer wants to be able to use Amazon CloudWatch to monitor and troubleshoot all applications from one place.
How can the developer accomplish this?
A. Install an AWS SDK on the on-premises server to automatically send logs to CloudWatch.
B. Download the CloudWatch agent to the on-premises server. Configure the agent to use IAM user credentials with permissions for CloudWatch.
C. Upload log files from the on-premises server to Amazon S3 and have CloudWatch read the files.
D. Upload log files from the on-premises server to an Amazon EC2 instance and have the instance forward the logs to CloudWatch.
Answer: (SHOW ANSWER)
Amazon CloudWatch is a service that monitors AWS resources and applications. The developer can use CloudWatch to monitor and troubleshoot all applications from one place. To do so, the developer needs to download the CloudWatch agent to the on-premises server and configure the agent to use IAM user credentials with permissions for CloudWatch. The agent will collect logs and metrics from the on-premises server and send them to CloudWatch.

References:
* [What Is Amazon CloudWatch? - Amazon CloudWatch]
* [Installing and Configuring the CloudWatch Agent - Amazon CloudWatch]

NEW QUESTION: 205
A developer is building a serverless application that uses asynchronous AWS Lambda functions. The developer needs a solution to capture records of every Lambda function invocation. Each function must have multiple destinations based on whether each invocation is successful. The solution must record function responses in JSON format. Which solution will meet these requirements?
A. Amazon CloudWatch Logs log groups that use the default log format for Lambda functions. Route each invocation to the appropriate log group by using a Lambda canary deployment and weighted aliases. Set the appropriate log group as the target for each function.
B. Set up an S3 bucket as an on-failure destination for the Lambda function. Configure an Amazon SNS topic as the destination for successful Lambda function invocations.
C. Configure an Amazon SQS dead-letter queue as an event source for the Lambda function to store failed invocations. In the Lambda function code, use the PutItem Amazon DynamoDB API call to add the successful invocation information to the database.
D. Set up an Amazon SQS queue as an on-failure destination for the Lambda function. Configure an Amazon OpenSearch Service cluster as the destination for the Lambda function for successful invocations.
Answer: (SHOW ANSWER)
For asynchronous Lambda invocations, AWS provides Lambda Destinations, which can route the result of every invocation to different targets based on outcome. Destinations support separate configuration for on- success and on-failure, and they send a structured payload that includes metadata and the function response (or error information). This payload is delivered as a JSON document, satisfying the requirement to record function responses in JSON format while also capturing every invocation record.
Option B matches this pattern directly: configure an on-failure destination (S3) and an on-success destination (SNS). With this configuration, each async invocation that succeeds results in a JSON record being delivered to the SNS topic, enabling downstream processing, notifications, or fan-out to other services.
Each async invocation that fails results in a JSON record being delivered to S3, which provides durable, cost- effective storage for later analysis or replay workflows. This design requires minimal code changes because the routing is handled by Lambda's destination configuration rather than custom logic in the function.
Option C relies on a DLQ for failures but then requires custom code to store success records in DynamoDB, increasing development effort and operational complexity. Option A is not aligned: canary deployments and log groups do not provide "multiple destinations per invocation outcome," and CloudWatch Logs do not natively separate success/failure

records as destinations with structured invocation payloads. Option D proposes OpenSearch as a success destination, but Lambda Destinations do not use OpenSearch directly as a destination target; the supported destination types are typically SQS, SNS, EventBridge, and Lambda, and you can store records in S3 through other integrations. Therefore, the best match that uses built-in destinations and records JSON invocation results is B.

NEW QUESTION: 206
An 1AM role is attached to an Amazon EC2 instance that explicitly denies access to all Amazon S3 API actions. The EC2 instance credentials file specifies the 1AM access key and secret access key, which allow full administrative access.
Given that multiple modes of 1AM access are present for this EC2 instance, which of the following is correct?
A. The EC2 instance will not be able to perform any S3 action on any S3 bucket.
B. The EC2 instance will be able to perform all actions on any S3 bucket.
C. The EC2 instance will only be able to list the S3 buckets.
D. The EC2 instance will only be able to list the contents of one S3 bucket at a time.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 207
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. Publish a new version of the Lambda function that contains the updated code.
B. Set up a new stage in API Gateway with a new Lambda function version. Enable weighted routing in API Gateway stages.
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. Set up a routing policy on a Network Load Balancer. Configure 10% of the traffic to go to the new Lambda function version.
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: (SHOW ANSWER)
Step 1: Understanding the Requirements
Gradual Traffic Shift: Test the new version by routing only 10% of production traffic to it.
Lambda Deployment: Use versioning and aliases to manage Lambda function updates.
Step 2: Solution Analysis
Option A:

Publishing a new version creates an immutable version of the Lambda function with the updated code.
This is a prerequisite for deploying changes using weighted aliases.
Correct option.
Option B:
API Gateway stages are not used for weighted routing; they represent environments like "dev" or "prod." Weighted routing is implemented using Lambda aliases, not API Gateway stages.
Not suitable.
Option C:
Lambda aliases allow traffic to be split between versions using weighted routing.
Assign a 90% weight to the old version and 10% to the new version to implement the gradual rollout.
Correct option.
Option D:
Network Load Balancers are not suitable for managing Lambda function traffic directly.
Not applicable.
Option E:
Route 53 routing policies apply at the DNS level and are not designed for Lambda version management.
Not suitable.
Step 3: Implementation Steps
Publish a New Version:
Publish the updated Lambda function code as a new version.
Create an Alias and Configure Weighted Routing:
Create an alias (e.g., prod) and associate it with both the old and new versions.
Set weights for traffic distribution:
aws lambda update-alias --function-name my-function \
--name prod --routing-config '{"AdditionalVersionWeights": {"2": 0.1}}' AWS Developer References:
Lambda Function Aliases
Weighted Traffic Routing with Lambda

NEW QUESTION: 208
A developer is creating a new application that will be accessed by users through an API created using Amazon API Gateway. The users need to be authenticated by a third-party Security Assertion Markup Language (SAML) identity provider. Once authenticated, users will need access to other AWS services, such as Amazon S3 and Amazon DynamoDB.
How can these requirements be met?
A. Use an Amazon Cognito user pool with SAML as the resource server.

B. Use Amazon Cognito identity pools with a SAML identity provider as one of the authentication providers.
C. Use the AWS IAM service to provide the sign-up and sign-in functionality.
D. Use Amazon CloudFront signed URLs to connect with the SAML identity provider.
Answer: B (LEAVE A REPLY)
The requirement has two parts: (1) authenticate users with a third-party SAML IdP, and (2) after authentication, allow those users to access AWS services like Amazon S3 and DynamoDB. The AWS pattern for this is to federate the external identity into AWS and then exchange that identity for temporary AWS credentials.
Amazon Cognito identity pools are designed to provide AWS credentials to authenticated users (via AWS STS) so the users can call AWS services directly or through an application backend. Identity pools support federation with external identity providers, including SAML 2.0. When a user authenticates with the third- party SAML IdP, the identity pool can accept the SAML assertion, map the user to an IAM role, and return temporary, scoped credentials (AccessKeyId/SecretAccessKey/SessionToken) associated with that role. Those credentials can be restricted using IAM policies to only the required S3 buckets, DynamoDB tables, and actions, satisfying least privilege.
Option A is incorrect because a Cognito user pool is primarily a user directory and OIDC/OAuth provider for application authentication; while user pools can integrate with SAML IdPs for federation, user pools alone do not directly issue AWS service credentials. The key requirement here is access to S3/DynamoDB, which is the role of an identity pool. Option C is not appropriate because IAM is not intended to provide end-user sign- up/sign-in for external users; it is for AWS identities and permissions. Option D is unrelated: CloudFront signed URLs control access to CloudFront-distributed content and do not authenticate users with SAML or provide AWS credentials.
Therefore, B meets both requirements: federate SAML authentication through a Cognito identity pool and provide temporary AWS credentials for accessing S3 and DynamoDB.

NEW QUESTION: 209
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 (LEAVE A REPLY)
Amazon S3 supports resource-based policies, which are JSON documents that specify the permissions for accessing S3 resources. A resource-based policy can be used to enforce encryption in transit by denying access to requests that do not use HTTPS. The condition key aws:SecureTransport can be used to check if the request was sent using SSL. If the value of this key is false, the request is denied; otherwise, the request is allowed.
Reference: How do I use an S3 bucket policy to require requests to use Secure Socket Layer (SSL)?

NEW QUESTION: 210
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.
Answer: B (LEAVE A REPLY)
The objects are "rarely accessed after 1 week" but must remain immediately available at all times. That means the storage class must support millisecond access without a restore process. S3 Standard-Infrequent Access (Standard-IA) is designed for exactly this: lower storage cost than S3 Standard, while still providing rapid access when needed.
With option B, an S3 Lifecycle rule transitions objects from S3 Standard to S3 Standard-IA after 7 days. This aligns perfectly with the usage pattern: frequent access initially, then infrequent access after a week, while keeping immediate availability.
Option C (S3 Glacier Flexible Retrieval) is not appropriate because Glacier classes are archival. Access typically requires a restore operation and retrieval time (minutes to hours), which violates "immediately available at all times." Option A expires objects, which deletes them after 7 days-this contradicts the requirement to keep objects available.
Option D is irrelevant because delete markers exist only when versioning is enabled. The bucket does not have versioning enabled, so this rule would not help.

Therefore, transitioning to S3 Standard-IA after 7 days is the correct cost-optimized solution while maintaining immediate availability.

NEW QUESTION: 211
A company has an application that consists of different microservices that run inside an AWS account. The microservices are running in containers inside a single VPC. The number of microservices is constantly increasing. A developer must create a central logging solution for application logs.
Which solution will meet these requirements?
A. Create a different Amazon CloudWatch Logs stream for each microservice.
B. Create an AWS CloudTrail trail to log all the API calls.
C. Configure VPC Flow Logs to track the communications between the microservices.
D. Use AWS Cloud Map to map the interactions of the microservices.
Answer: A (LEAVE A REPLY)
Centralized logging for microservices means collecting application logs from many services into a single managed logging backend where logs can be searched, retained, and monitored. For containerized microservices on AWS, the standard approach is to send each service's stdout/stderr logs to Amazon CloudWatch Logs.
Option A is the appropriate solution: create separate CloudWatch Logs streams (and typically log groups) per microservice. This scales naturally as microservices increase, supports retention policies, metric filters, alarms, and integrations with tools like CloudWatch Logs Insights for querying across services. Container orchestrators (ECS/EKS) can be configured to use the CloudWatch Logs log driver so logs are shipped automatically without custom log shipping agents.
Option B (CloudTrail) logs AWS API activity, not application logs.
Option C (VPC Flow Logs) captures network flow metadata, not application-level logs.
Option D (Cloud Map) is service discovery, not logging.
Therefore, using CloudWatch Logs streams per microservice is the correct central logging approach.

NEW QUESTION: 212

A developer is creating an AWS Lambda function that needs network access to private resources in a VPC.
A. Attach the Lambda function to the VPC through private subnets. Create a security group that allows network access to the private resources. Associate the security group with the Lambda function.
B. Configure the Lambda function to route traffic through a VPN connection. Create a security group that allows network access to the private resources. Associate the security group with the Lambda function.
C. Configure a VPC endpoint connection for the Lambda function. Set up the VPC endpoint to route traffic through a NAT gateway.
D. Configure an AWS PrivateLink endpoint for the private resources. Configure the Lambda function to reference the PrivateLink endpoint.
Answer: (SHOW ANSWER)
Comprehensive Detailed Step by Step Explanation with All AWS Developer References:
When you need to provide an AWS Lambda function access to private resources in a VPC, the most common and straightforward approach is to attach the Lambda function to a VPC via private subnets. Once the Lambda function is associated with the VPC, you need to configure appropriate security groups to control the access to the private resources.
* Lambda with VPC Access: Lambda functions can be attached to private subnets in a VPC, allowing them to access resources like RDS, EC2, or internal services within that VPC.
* Security Groups: A security group acts as a virtual firewall for the Lambda function, ensuring that it can access only the necessary resources and ports in the VPC.
* Alternatives:
* Option B involves routing traffic through a VPN, which adds unnecessary complexity and operational overhead compared to simply attaching the Lambda to the VPC.
* Option C requires configuring a VPC endpoint and a NAT gateway, which can be complex and costly.
* Option D refers to AWS PrivateLink, which is used to access services over private connections, but it's unnecessary in this scenario unless you need a cross-VPC connection.
Lambda functions in a VPC

NEW QUESTION: 213
A company has multiple Amazon VPC endpoints in the same VPC. A developer needs configure an Amazon S3 bucket policy so users can access an S3 bucket only by using these VPC endpoints.
Which solution will meet these requirements?
A. Create multiple S3 bucket polices by using each VPC endpoint ID that have the aws SourceVpce value in the StringNotEquals condition.

B. Create a single S3 bucket policy that has the aws SourceVpc value and in the StingNotEquals condition to use VPC ID.
C. Create a single S3 bucket policy that the multiple aws SourceVpce value and in the SringNotEquals condton to use vpce.
D. Create a single S3 bucket policy that has multiple aws sourceVpce value in the StingNotEquale condition. Repeat for all the VPC endpoint IDs.
Answer: D (LEAVE A REPLY)
This solution will meet the requirements by creating a single S3 bucket policy that denies access to the S3 bucket unless the request comes from one of the specified VPC endpoints. The aws:SourceVpce condition key is used to match the ID of the VPC endpoint that is used to access the S3 bucket. The StringNotEquals condition operator is used to negate the condition, so that only requests from the listed VPC endpoints are allowed. Option A is not optimal because it will create multiple S3 bucket policies, which is not possible as only one bucket policy can be attached to an S3 bucket. Option B is not optimal because it will use the aws:
SourceVpc condition key, which matches the ID of the VPC that is used to access the S3 bucket, not the VPC endpoint. Option C is not optimal because it will use the StringNotEquals condition operator with a single value, which will deny access to the S3 bucket from all VPC endpoints except one.
References: Using Amazon S3 Bucket Policies and User Policies, AWS Global Condition Context Keys

NEW QUESTION: 214
A company is migrating an on-premises database to Amazon RDS for MySQL. The company has read-heavy workloads. The company wants to refactor the code to achieve optimum read performance for queries.
Which solution will meet this requirement with LEAST current and future effort?
A. Use a multi-AZ Amazon RDS deployment. Increase the number of connections that the code makes to the database or increase the connection pool size if a connection pool is in use.
B. Use a multi-AZ Amazon RDS deployment. Modify the code so that queries access the secondary RDS instance.
C. Deploy Amazon RDS with one or more read replicas. Modify the application code so that queries use the URL for the read replicas.
D. Use open source replication software to create a copy of the MySQL database on an Amazon EC2 instance. Modify the application code so that queries use the IP address of the EC2 instance.
Answer: C (LEAVE A REPLY)
Amazon RDS for MySQL supports read replicas, which are copies of the primary database instance that can handle read-only queries. Read replicas can improve the read performance of the database by offloading the read workload from the primary instance

and distributing it across multiple replicas. To use read replicas, the application code needs to be modified to direct read queries to the URL of the read replicas, while write queries still go to the URL of the primary instance. This solution requires less current and future effort than using a multi-AZ deployment, which does not provide read scaling benefits, or using open source replication software, which requires additional configuration and maintenance. Reference: Working with read replicas

NEW QUESTION: 215
A company is building a compute-intensive application that will run on a fleet of Amazon EC2 instances. The application uses attached Amazon Elastic Block Store (Amazon EBS) volumes for storing dat a. The Amazon EBS volumes will be created at time of initial deployment. The application will process sensitive information. All of the data must be encrypted. The solution should not impact the application's performance.
Which solution will meet these requirements?
A. Configure the fleet of EC2 instances to use encrypted EBS volumes to store data.
B. Configure the application to write all data to an encrypted Amazon S3 bucket.
C. Configure a custom encryption algorithm for the application that will encrypt and decrypt all data.
D. Configure an Amazon Machine Image (AMI) that has an encrypted root volume and store the data to ephemeral disks.
Answer: A (LEAVE A REPLY)
Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with Amazon EC2 instances1. Amazon EBS encryption offers a straight-forward encryption solution for your EBS resources associated with your EC2 instances1. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted: Data at rest inside the volume, all data moving between the volume and the instance, all snapshots created from the volume, and all volumes created from those snapshots1. Therefore, option A is correct.

NEW QUESTION: 216
A developer is working on a Python application that runs on Amazon EC2 instances. The developer wants to enable tracing of application requests to debug performance issues in the code.
Which combination of actions should the developer take to achieve this goal? (Select TWO)
A. Install the Amazon CloudWatch agent on the EC2 instances.
B. Install the AWS X-Ray daemon on the EC2 instances.
C. Configure the application to write JSON-formatted togs to /var/log/cloudwatch.
D. Configure the application to write trace data to /Var/log-/xray.
E. Install and configure the AWS X-Ray SDK for Python in the application.
Answer: B,E (LEAVE A REPLY)

This solution will meet the requirements by using AWS X-Ray to enable tracing of application requests to debug performance issues in the code. AWS X-Ray is a service that collects data about requests that the applications serve, and provides tools to view, filter, and gain insights into that data. The developer can install the AWS X-Ray daemon on the EC2 instances, which is a software that listens for traffic on UDP port 2000, gathers raw segment data, and relays it to the X-Ray API. The developer can also install and configure the AWS X-Ray SDK for Python in the application, which is a library that enables instrumenting Python code to generate and send trace data to the X-Ray daemon. Option A is not optimal because it will install the Amazon CloudWatch agent on the EC2 instances, which is a software that collects metrics and logs from EC2 instances and on-premises servers, not application performance data. Option C is not optimal because it will configure the application to write JSON-formatted logs to /var/log/cloudwatch, which is not a valid path or destination for CloudWatch logs. Option D is not optimal because it will configure the application to write trace data to /var/log/xray, which is also not a valid path or destination for X-Ray trace data.
References: [AWS X-Ray], [Running the X-Ray Daemon on Amazon EC2]

NEW QUESTION: 217
An application writes transactions to an Amazon DynamoDB table by using the PutItem operation. Each transaction has a unique transactionId. Sometimes duplicate transactions are received. The developer wants to ensure that a duplicate PutItem does not overwrite an existing item. Duplicate transactions are rare.
What is the MOST cost-effective solution?
A. Call GetItem before calling PutItem.
B. Enable TTL on the table.
C. Use a conditional put with attribute_exists(transactionId).
D. Use a conditional put with attribute_not_exists(transactionId).
Answer: D (LEAVE A REPLY)
Amazon DynamoDB supports conditional writes, which allow developers to enforce data integrity rules directly at the database layer. AWS documentation states that conditional expressions are the recommended way to prevent accidental overwrites and ensure idempotent writes.
Using a PutItem operation with the condition expression attribute_not_exists(transactionId) ensures that the write succeeds only if the item does not already exist. If a duplicate transaction is received, DynamoDB rejects the request without modifying the existing record.
This approach is highly cost-effective because it avoids additional read operations. Option A doubles request cost by performing a read before every write. TTL (Option B) is unrelated to overwrite prevention. Option C does the opposite of the requirement by ensuring the attribute exists.

AWS explicitly recommends conditional writes for handling duplicates and enforcing uniqueness with minimal cost and latency.
Therefore, implementing a conditional put with attribute_not_exists(transactionId) is the correct solution.

NEW QUESTION: 218
A developer is writing an AWS Lambda function. The developer wants to log key events that occur while the Lambda function runs. The developer wants to include a unique identifier to associate the events with a specific function invocation. The developer adds the following code to the Lambda function:

```
function handler(event, context) {

}
```

Which solution will meet this requirement?
A. Obtain the request identifier from the AWS request ID field in the context object. Configure the application to write logs to standard output.
B. Obtain the request identifier from the AWS request ID field in the event object. Configure the application to write logs to a file.
C. Obtain the request identifier from the AWS request ID field in the event object. Configure the application to write logs to standard output.
D. Obtain the request identifier from the AWS request ID field in the context object. Configure the application to write logs to a file.
Answer: A (LEAVE A REPLY)
https://docs.aws.amazon.com/lambda/latest/dg/nodejs-context.html
https://docs.aws.amazon.com/lambda
/latest/dg/nodejs-logging.html
There is no explicit information for the runtime, the code is written in Node.js.
AWS Lambda is a service that lets developers run code without provisioning or managing servers. The developer can use the AWS request ID field in the context object to obtain a unique identifier for each function invocation. The developer can configure the application to write logs to standard output, which will be captured by Amazon CloudWatch Logs. This solution will meet the requirement of logging key events with a unique identifier.
References:
[What Is AWS Lambda? - AWS Lambda]
[AWS Lambda Function Handler in Node.js - AWS Lambda]
[Using Amazon CloudWatch - AWS Lambda]

NEW QUESTION: 219
A developer is setting up infrastructure by using AWS Cloud Formation. If an error occurs when the resources described in the CloudFormation template are provisioned,

successfully provisioned resources must be preserved. The developer must provision and update the CloudFormation stack by using the AWS CLI.
Which solution will meet these requirements?
A. Add a -disable-roll back command line option to the create-stack command and the update-stack command
B. Add a -parameters ParameterKey=P reserve Resources. ParameterValue=True command line option to the create-stack command and the update-stack command.
C. Add an --enable-terminal ion-protection command line option to the create-stack command and the update-stack command.
D. Add a -tags Key=PreserveResources.Value=True command line option to the create-stack command and the update-stack command.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 220
A company needs to rapidly prototype a web application. However, the company has not yet designed the complete architecture.
A developer uses AWS Lambda functions to build three endpoints. A frontend team wants to test the endpoints while the team prototypes the frontend.
Which solution will meet these requirements with the LEAST operational overhead?
A. Set up a Lambda function URL for each endpoint. Use the function URLs for testing.
B. Set up an Amazon API Gateway REST API to have a Lambda proxy integration. Use the REST API endpoint URL for testing.
C. Set up an AWS AppSync API to have a Lambda resolver. Use a GraphQL endpoint for testing.
D. Set up an Amazon ECS container that runs an open source web proxy and Lambda code. Use the web proxy endpoint for testing.
Answer: A (LEAVE A REPLY)
For rapid prototyping with the least operational overhead, Lambda function URLs are the best fit. A Lambda function URL provides a built-in HTTPS endpoint directly in front of a Lambda function, allowing the frontend team to invoke each endpoint without requiring additional infrastructure. This is ideal when the overall architecture is not finalized and the team needs a quick way to test.
AWS describes Lambda function URLs as a simple way to "add an HTTPS endpoint to your Lambda function" without requiring API Gateway, load balancers, or custom proxy layers. This substantially reduces setup time and avoids the operational tasks associated with provisioning, configuring, and managing an API layer during early-stage development.
Option B (API Gateway REST API) introduces more operational overhead: creating resources, methods, integrations, deployments, stages, and potentially authorization and throttling configuration. API Gateway is powerful, but it is more than needed for quick endpoint testing.

Option C (AppSync) is designed for GraphQL-based APIs and requires schema design and resolvers. That is unnecessary complexity for simply testing three Lambda-backed endpoints.
Option D (ECS proxy) adds the most operational burden because it requires container orchestration, networking, scaling, and patching-completely misaligned with "least operational overhead." Therefore, the fastest and simplest approach is to create a function URL per Lambda endpoint, allowing immediate testing from the frontend with minimal additional AWS configuration.

NEW QUESTION: 221
A company has a web application that contains an Amazon API Gateway REST API. A developer has created an AWS CloudFormation template for the initial deployment of the application. The developer has deployed the application successfully as part of an AWS CodePipeline CI/CD process. All resources and methods are available through the deployed stage endpoint.
The CloudFormation template contains the following resource types:
* AWS::ApiGateway::RestApi
* AWS::ApiGateway::Resource
* AWS::ApiGateway::Method
* AWS::ApiGateway::Stage
* AWS::ApiGateway::Deployment
The developer adds a new resource to the REST API with additional methods and redeploys the template.
CloudFormation reports that the deployment is successful and that the stack is in the UPDATE_COMPLETE state. However, calls to all new methods are returning 404 (Not Found) errors.
What should the developer do to make the new methods available?
A. Specify the disable-rollback option during the update-stack operation.
B. Unset the CloudFormation stack failure options.
C. Add an AWS CodeBuild stage to CodePipeline to run the aws apigateway create-deployment AWS CLI command.
D. Add an action to CodePipeline to run the aws cloudfront create-invalidation AWS CLI command.
Answer: (SHOW ANSWER)
For API Gateway REST APIs, configuration changes (new resources/methods) do not automatically become callable from a stage until a new API Gateway Deployment is created and associated with the stage. In CloudFormation, an AWS::ApiGateway::Deployment resource is immutable and effectively represents a snapshot of the API configuration at a point in time. If the Deployment resource does not change (for example, its logical ID or a property that forces replacement), CloudFormation may not create a new deployment even though the stack update succeeds-resulting in the

stage still pointing to the old deployment. The symptom is exactly what's described: new methods return 404 because they are not part of the deployed snapshot.
A common operational fix is to create a new deployment explicitly during CI/CD. Option C does this by running aws apigateway create-deployment, which creates a fresh deployment that includes the new resources
/methods and makes them available on the stage.
Option D is unrelated (CloudFront invalidation).
Options A and B do not affect API Gateway deployment snapshots.
In pure CloudFormation, another common pattern is to force a new deployment by changing the Deployment logical ID or embedding a timestamp/hash in the deployment description. But among the options given, running create-deployment is the direct fix.
Therefore, add a pipeline step to run aws apigateway create-deployment.

NEW QUESTION: 222
A company runs an ecommerce application on AWS. The application stores data in an Amazon Aurora database.
A developer is adding a caching layer to the application. The caching strategy must ensure that the application always uses the most recent value for each data item.
Which caching strategy will meet these requirements?
A. Implement a read-through strategy for every item that is loaded.
B. Implement a TTL strategy for every item that is saved in the cache.
C. Implement a lazy loading strategy for every item that is loaded.
D. Implement a write-through strategy for every item that is created and updated.
Answer: (SHOW ANSWER)

NEW QUESTION: 223
A company is building an application to accept data from customers. The data must be encrypted at rest and in transit.
The application uses an Amazon API Gateway API that resolves to AWS Lambda functions. The Lambda functions store the data in an Amazon Aurora MySQL DB cluster.
The application worked properly during testing.
A developer configured an Amazon CloudFront distribution with field-level encryption that uses an AWS Key Management Service (AWS KMS) key. After the configuration of the distribution, the application behaved unexpectedly. All the data in the database changed from plaintext to ciphertext.
The developer must ensure that the data is not stored in the database as the ciphertext from the CloudFront field-level encryption.
Which solution will meet this requirement?
A. Add a Lambda function that uses the KMS key to decrypt the data fields before saving the data to the database.

B. Enable encryption on the DB cluster by using the same KMS key that is used in CloudFront.
C. Change the CloudFront Viewer protocol policy from "HTTP and HTTPS" to "HTTPS only."
D. Request and deploy a new SSL certificate to use with the CloudFront distribution.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 224
A developer is building various microservices for an application that will run on Amazon EC2 instances. The developer needs to monitor the end-to-end view of the requests between the microservices and debug any issues in the various microservices.
What should the developer do to accomplish these tasks?
A. Use Amazon CloudWatch to aggregate the microservices' logs and metrics, and build the monitoring dashboard.
B. Use AWS CloudTrail to aggregate the microservices' logs and metrics, and build the monitoring dashboard.
C. Use the AWS X-Ray SDK to add instrumentation in all the microservices, and monitor using the X-Ray service map.
D. Use AWS Health to monitor the health of all the microservices.
Answer: C (LEAVE A REPLY)
To monitor end-to-end requests and debug issues across microservices, you need distributed tracing. Among AWS tools, AWS X-Ray is specifically built for this use case.
Step-by-Step Breakdown:
Step 1: Understand the Requirement
The developer needs:
End-to-end request tracing across microservices
Debugging capability for performance issues or failures
This points directly to a distributed tracing solution rather than just logs or metrics.
Step 2: Evaluate the Options
Option A: Amazon CloudWatch
CloudWatch is powerful for metrics, logs, and alerts.
But it does not provide distributed tracing or request path visualization between microservices.
So, it's not sufficient alone for the requirement.
Option B: AWS CloudTrail
CloudTrail tracks API calls made through the AWS Management Console, CLI, SDKs.
It is meant for auditing and governance, not microservices request tracing.
Not suitable for debugging microservices interactions.
Option C: AWS X-Ray SDK with X-Ray service map
Purpose-built for distributed tracing

Automatically collects data about requests as they travel through your microservices You can instrument your code with the AWS X-Ray SDK in Java, Node.js, Python, Go, .NET, etc.
Visualizes requests using a service map, showing latency, errors, and time spent in downstream services.
How it works:
Instrument each microservice using the AWS X-Ray SDK.
Use the SDK to trace requests, propagate trace headers across services.
The X-Ray daemon collects and sends trace data to the X-Ray service.
You can view the service map, see bottlenecks, error rates, etc.
Option D: AWS Health
AWS Health provides information on AWS service outages or account-level events.
It does not monitor your application or microservices health or request flows.
Correct Choice: C is the only option that meets the developer's requirement to monitor end-to-end request paths and debug issues in a microservices architecture.
AWS Developer References:
AWS X-Ray Documentation: https://docs.aws.amazon.com/xray/latest/devguide/aws-xray.html Instrumenting your application: https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk.html Viewing the Service Map: https://docs.aws.amazon.com/xray/latest/devguide/xray-console-service-map.html Tracing Microservices with AWS X-Ray: https://aws.amazon.com/blogs/compute/tracing-microservices-aws- x-ray/

NEW QUESTION: 225
A developer is building an application that processes a stream of user-supplied data. The data stream must be consumed by multiple Amazon EC2-based processing applications in parallel and in real time. Each processor must be able to resume without losing data if there is a service interruption. The application architect plans to add other processors in the near future and wants to minimize the amount of data duplication involved.
Which solution will satisfy these requirements?
A. Publish the data to Amazon SQS.
B. Publish the data to Amazon Data Firehose.
C. Publish the data to Amazon EventBridge.
D. Publish the data to Amazon Kinesis Data Streams.
Answer: D (LEAVE A REPLY)
The requirements describe a real-time streaming use case where multiple processing applications must read the same stream in parallel, each must be able to resume processing after interruptions without data loss, and the system should support adding new processors with minimal duplication. Amazon Kinesis Data Streams (KDS) is purpose-built for this pattern.

With Kinesis Data Streams, producers write records to a stream that is split into shards. Multiple consumer applications can independently read from the stream using separate consumer groups (for example, using Kinesis Client Library). Each consumer maintains its own checkpoint (sequence number position), which allows it to resume from the last processed record after a restart or failure. This directly meets the "resume without losing data" requirement.
KDS also supports a configurable retention period (commonly 24 hours by default, extendable), enabling replay and recovery without forcing producers to re-send data or duplicating messages for each processor.
Adding new consumers is straightforward: a new processor can begin reading the existing stream (from latest or from a trim horizon), which minimizes duplication compared to fan-out duplication patterns.
Why the others don't fit:
* SQS is a queue, not a stream. Multiple consumers typically compete for messages rather than all consuming the same messages independently, and duplication is required if multiple apps must process the same data.
* Kinesis Data Firehose is primarily for delivery to destinations (S3, Redshift, OpenSearch) and not designed for multiple real-time parallel consumers with replay semantics.
* EventBridge is event routing with limited replay/resume semantics compared to KDS and is not intended for high-throughput stream processing with per-consumer checkpoints.
Therefore, Amazon Kinesis Data Streams satisfies all requirements.

NEW QUESTION: 226
A developer has created an AWS Lambda function that is written in Python. The Lambda function reads data from objects in Amazon S3 and writes data to an Amazon DynamoDB table.
The function is successfully invoked from an S3 event notification when an object is created. However, the function fails when it attempts to write to the DynamoDB table.
What is the MOST likely cause of this issue?
A. The Lambda function's concurrency limit has been exceeded.
B. The DynamoDB table requires a global secondary index (GSI) to support writes.
C. The Lambda function does not have IAM permissions to write to DynamoDB.
D. The DynamoDB table is not running in the same Availability Zone as the Lambda function.
Answer: C (LEAVE A REPLY)
Because the Lambda function is successfully triggered by the S3 event notification, the invocation path (S3 # Lambda) is working correctly. The failure occurs specifically when the function tries to write to DynamoDB, which strongly indicates an authorization problem rather than an invocation, scaling, or infrastructure issue.
In AWS, a Lambda function interacts with other services by using its execution role (an IAM role). AWS documentation explains that a Lambda function must have explicit IAM

permissions to call downstream services such as DynamoDB. To write items, the role typically needs actions like dynamodb:PutItem (and sometimes dynamodb:UpdateItem, dynamodb:BatchWriteItem, depending on code behavior) on the target table resource ARN. If these permissions are missing or scoped incorrectly, DynamoDB returns an AccessDeniedException (or similar) and the function fails at the write step.
Option A is unlikely because exceeding concurrency would typically prevent invocation or cause throttling at the Lambda service level, not selectively fail only at DynamoDB write time after the function begins executing.
Option B is incorrect: DynamoDB does not require a GSI to support writes. GSIs are for alternate query access patterns, not mandatory for write operations.
Option D is incorrect because DynamoDB is a regional service, not tied to a single Availability Zone, and Lambda does not need to be "in the same AZ" to access it.
Therefore, the most likely cause is that the Lambda execution role lacks the necessary IAM permissions to perform DynamoDB write operations.

NEW QUESTION: 227
A developer is monitoring an application that runs on an Amazon EC2 Instance. The developer has configured a custom Amazon CloudWatch metric with data granularity of 1 second. It any issues occur, the developer wants to be notified within 30 seconds by Amazon Simple Notification Service (Amazon SNS).
What should the developer do to meet this requirement?
A. Set up a custom CloudWatch dashboard.
B. Use Amazon CloudWatch Logs Insights.
C. Change to a default CloudWatch metric.
D. Configure a high-resolution CloudWatch alarm.
Answer: D (LEAVE A REPLY)

NEW QUESTION: 228
A developer is working on an ecommerce application that stores data in an Amazon RDS for MySQL cluster.
The developer needs to implement a caching layer for the application to retrieve information about the most viewed products.

Which solution will meet these requirements?

A. Edit the RDS for MySQL cluster by adding a cache node. Configure the cache endpoint instead of the cluster endpoint in the application.

B. Create an Amazon ElastiCache (Redis OSS) cluster. Update the application code to use the ElastiCache (Redis OSS) cluster endpoint.

C. Create an Amazon DynamoDB Accelerator (DAX) cluster in front of the RDS for MySQL cluster.

Configure the application to connect to the DAX endpoint instead of the RDS endpoint.

D. Configure the RDS for MySQL cluster to add a standby instance in a different Availability Zone.Configure the application to read the data from the standby instance.

Answer: B (LEAVE A REPLY)

To add a caching layer for frequently requested data (such as "most viewed products"), the AWS-native choice is Amazon ElastiCache, commonly using Redis for low-latency key/value caching, counters, sorted sets, and leaderboard-like patterns. Redis is particularly well-suited for ecommerce "top N" or popularity queries because it supports atomic increments and sorted sets, enabling efficient ranking updates.

Option B correctly introduces an ElastiCache (Redis OSS) cluster and updates the application to read from Redis first (cache-aside or write-through patterns) instead of hitting the RDS MySQL cluster for every request. This offloads read pressure from the database, reduces latency, and improves overall throughput.

Option A is not a valid RDS feature: you do not add a "cache node" to an RDS MySQL cluster as part of RDS itself.

Option C is incorrect because DAX is a caching layer specifically for DynamoDB, not for RDS MySQL.

Option D improves availability via Multi-AZ standby, but the standby is not for read scaling in standard RDS Multi-AZ (it's primarily for failover). It does not provide a cache and does not solve the performance needs for hot reads.

NEW QUESTION: 229

A retail company runs a sales analytics application that uses an AWS Lambda function to process transaction data that is stored in Amazon DocumentDB. The application aggregates daily sales data across 500 stores and uses the data to generate reports for senior managers.

Application users report that the application is taking longer to generate reports and that their requests sometimes time out. A developer investigates and notices that the application's average response time for report generation has increased from 3 seconds to over 25 seconds.

The developer needs to identify the application's performance bottlenecks.

Which solution will meet this requirement with the LEAST operational overhead?

A. Enable AWS X-Ray tracing for the Lambda function and DocumentDB cluster. Implement custom subsegments to track query execution to identify slow-performing queries.
B. Add Amazon CloudWatch Logs error streaming. Create custom CloudWatch metrics based on the logs.
Create a CloudWatch dashboard that shows Lambda metrics.
C. Modify the Lambda function to use DocumentDB connection pooling. Implement async/await patterns for database operations.
D. Add logging statements within the Lambda function to output query execution times and database connection attempts. Store IDs in Amazon CloudWatch Logs. Use CloudWatch Logs Insights to analyze the logs.
Answer: A (LEAVE A REPLY)
The goal is to identify performance bottlenecks across a Lambda function that calls a database (DocumentDB). The most operationally efficient way to pinpoint where time is being spent-without building custom logging/metrics pipelines-is to use distributed tracing. AWS X-Ray provides end-to-end request traces, timing breakdowns, and service maps that help identify whether latency is dominated by Lambda initialization, downstream calls, or database queries.
By enabling X-Ray tracing on the Lambda function, the developer can capture traces for report-generation requests and examine segments that show duration spent inside the function and in downstream dependencies.
To specifically diagnose database time, the developer can add custom subsegments around DocumentDB operations (query execution, connection acquisition, cursor iteration). This produces precise timing data for each step, making it straightforward to identify slow queries, excessive connection setup time, or serialization overhead. Because X-Ray aggregates and visualizes traces, the developer can quickly compare "fast" and "slow" traces and isolate the bottleneck.
Option B focuses on errors and generic metrics dashboards; it's useful for monitoring, but it won't precisely attribute the extra 22+ seconds to a particular downstream call path. Option C proposes performance improvements (pooling/async), but the question asks to identify bottlenecks first, and it also requires code changes and validation without confirming the root cause. Option D requires adding detailed logging statements and then querying logs; this is more development effort and more ongoing log volume/cost than turning on X-Ray and using trace analysis, especially when the intent is bottleneck identification rather than permanent instrumentation.
Therefore, A is the best choice: enable AWS X-Ray for the Lambda function and trace DocumentDB interactions with custom subsegments to quickly and efficiently identify the specific source(s) of increased latency.

NEW QUESTION: 230

A developer is building a new application on AWS. The application uses an AWS Lambda function that retrieves information from an Amazon DynamoDB table. The developer hardcoded the DynamoDB table name into the Lambda function code. The table name might change over time. The developer does not want to modify the Lambda code if the table name changes.
Which solution will meet these requirements MOST efficiently?
A. Create a Lambda environment variable to store the table name. Use the standard method for the programming language to retrieve the variable.
B. Store the table name in a file. Store the file in the /tmp folder. Use the SDK for the programming language to retrieve the table name.
C. Create a file to store the table name. Zip the file and upload the file to the Lambda layer. Use the SDK for the programming language to retrieve the table name.
D. Create a global variable that is outside the handler in the Lambda function to store the table name.
Answer: A (LEAVE A REPLY)
The simplest and most efficient way to avoid hardcoding configuration such as a DynamoDB table name is to use Lambda environment variables. Environment variables are designed for runtime configuration and allow changing values without updating application code logic. The developer can store the table name in an environment variable (for example, TABLE_NAME) and read it from the runtime environment using the standard language method (such as os.environ in Python, process.env in Node.js, etc.).
This approach has very low operational overhead: updating the table name becomes a configuration change (in the Lambda console, IaC templates, or CI/CD pipeline) rather than a code change. It also keeps deployment packages stable and avoids unnecessary dependencies.
Option B is not suitable because /tmp is ephemeral storage that can be cleared between invocations and is not intended for configuration management.
Option C is heavier than necessary. Lambda layers are great for shared libraries and dependencies, but using a layer to store a single changing table name is awkward and forces a layer update and function reconfiguration when the value changes.
Option D does not solve the problem because a global variable is still set in the code. If the table name changes, the code must still be modified and redeployed.
Therefore, storing the table name in a Lambda environment variable is the most efficient solution.

NEW QUESTION: 231
A company runs a web application on Amazon EC2 instances behind an Application Load Balancer. The application uses Amazon DynamoDB as its database. The company wants to ensure high performance for reads and writes.
Which solution will meet these requirements MOST cost-effectively?

A. Configure auto-scaling for the DynamoDB table with a target utilization of 70%. Set the minimum and maximum capacity units based on the expected workload.
B. Use DynamoDB on-demand capacity mode for the table. Specify a maximum throughput higher than the expected peak read and write capacity units.
C. Use DynamoDB provisioned throughput mode for the table. Create an Amazon CloudWatch alarm on the ThrottledRequests metric. Invoke an AWS Lambda function to increase provisioned capacity.
D. Create an Amazon DynamoDB Accelerator (DAX) cluster. Configure the application to use the DAX endpoint.
Answer: A (LEAVE A REPLY)
Why Option A is Correct:Auto-scaling with a target utilization ensures the DynamoDB table dynamically adjusts capacity based on workload, maintaining high performance while optimizing cost. Setting a reasonable target utilization minimizes overprovisioning and throttling risks.
Why Other Options are Incorrect:
Option B: On-demand capacity is costlier than provisioned throughput for predictable workloads.
Option C: Using manual CloudWatch alarms and Lambda for scaling is less efficient and adds operational overhead.
Option D: DAX accelerates read performance but does not improve write performance.
AWS Documentation References:
DynamoDB Auto Scaling

NEW QUESTION: 232
A developer is deploying an AWS Lambda function The developer wants the ability to return to older versions of the function quickly and seamlessly.
How can the developer achieve this goal with the LEAST operational overhead?
A. Use AWS OpsWorks to perform blue/green deployments.
B. Use a function alias with different versions.
C. Maintain deployment packages for older versions in Amazon S3.
D. Use AWS CodePipeline for deployments and rollbacks.
Answer: (SHOW ANSWER)
A function alias is a pointer to a specific Lambda function version. You can use aliases to create different environments for your function, such as development, testing, and production. You can also use aliases to perform blue/green deployments by shifting traffic between two versions of your function gradually. This way, you can easily roll back to a previous version if something goes wrong, without having to redeploy your code or change your configuration. Reference: AWS Lambda function aliases

NEW QUESTION: 233

A company runs an application in a third-party cloud. The company wants to use the application to update data in AWS by using API calls to AWS services. The API calls require credentials.
The company's security policy requires the company to limit the scope and duration of any credentials used to make API calls to AWS services.
Which solution will meet these requirements in the MOST secure way?
A. Create an IAM user for the application. Configure the application to load the IAM user's credentials as environment variables. Use the IAM user's credentials to interact with AWS services.
B. Create an IAM user for the application. Populate an AWS Secrets Manager secret with the IAM user's AWS credentials. Use the secret to interact with AWS services.
C. Create an IAM role for the application. Configure the application to call the AWS STS GetFederationToken API. Use the STS credentials to interact with AWS services.
D. Create an IAM role for the application. Configure the application to call the AWS STS AssumeRole API. Use the STS credentials to interact with AWS services.
Answer: D (LEAVE A REPLY)
The key security requirement is to limit both scope and duration of credentials used by an external application (running outside AWS). The most secure AWS-native way to do this is to use temporary security credentials issued by AWS Security Token Service (STS), rather than long-term IAM user access keys. Temporary credentials have a short, configurable lifetime and are tied to permissions defined by an IAM role and (optionally) session policies, which enforces least privilege.
With STS AssumeRole, the application requests temporary credentials for a specific IAM role. The role's permission policy strictly defines what AWS actions and resources the session can access. The resulting credentials automatically expire, reducing the blast radius if the credentials are exposed. This approach also supports best practices such as rotating session credentials frequently and using external IDs and condition keys (where applicable) to reduce confused-deputy risks.
Options A and B rely on long-term IAM user credentials. Even if stored in environment variables or AWS Secrets Manager, these are still persistent credentials that do not inherently meet the "limit duration" requirement and are higher risk if leaked. Secrets Manager improves storage and rotation workflows, but it does not change the fact that IAM user access keys are long-lived by default.
Option C (GetFederationToken) is not the best fit here. Federation tokens are typically used to obtain temporary credentials for a federated user session and are commonly associated with IAM users (or scenarios like providing temporary access to third parties) rather than the standard, role-based pattern for an application assuming permissions. The most direct and widely recommended method for applications needing scoped, time-bound AWS access is AssumeRole.

Therefore, D is the most secure solution: create an IAM role with least-privilege permissions and have the application call STS AssumeRole to obtain short-lived credentials for AWS API calls.

NEW QUESTION: 234
A developer is updating several AWS Lambda functions and notices that all the Lambda functions share the same custom libraries. The developer wants to centralize all the libraries, update the libraries in a convenient way, and keep the libraries versioned.
Which solution will meet these requirements with the LEAST development effort?
A. Create an AWS CodeArtifact repository that contains all the custom libraries.
B. Create a custom container image for the Lambda functions to save all the custom libraries.
C. Create a Lambda layer that contains all the custom libraries.
D. Create an Amazon EFS file system to store all the custom libraries.
Answer: C (LEAVE A REPLY)
AWS Lambda layers are specifically designed to share common code and dependencies across multiple Lambda functions. A layer is a ZIP archive that contains libraries, a custom runtime, or other function dependencies. Layers are versioned, so the developer can publish a new layer version when libraries change and then update functions to reference the new version. This directly meets the requirements to centralize libraries, update conveniently, and keep libraries versioned-while requiring minimal development work.
Using a single shared layer significantly reduces duplication across function deployment packages. It also simplifies CI/CD because the developer can build and publish the layer independently and reuse it across many functions.
Option A (CodeArtifact) is useful for dependency management, but it still requires each Lambda deployment to package dependencies or download them during build. It's more moving parts than a layer for this use case.
Option B (container images) can centralize dependencies but typically requires more effort: building, scanning, versioning container images, and updating function image URIs. It's heavier operationally than layers for "shared custom libraries." Option D (EFS) can be mounted by Lambda, but introduces networking considerations (VPC config), EFS lifecycle/permissions, and is unnecessary for simply sharing libraries. Also, cold start and operational overhead can increase.
Therefore, creating a Lambda layer for the shared libraries is the least-effort, most standard approach.

NEW QUESTION: 235
A company is planning to deploy an application on AWS behind an Elastic Load Balancing (ELB) load balancer. The application uses an HTTP/HTTPS listener and must access the client IP addresses.
Which load-balancing solution meets these requirements?

A. Use a Network Load Balancer (NLB). Enable proxy protocol support on the NLB and the target application.
B. Use an Application Load Balancer and the X-Forwarded-For headers.
C. Use an Application Load Balancer. Register the targets by the instance ID.
D. Use a Network Load Balancer and the X-Forwarded-For headers.
Answer: (SHOW ANSWER)

NEW QUESTION: 236
A developer is investigating recent performance bottlenecks within a company's distributed web application that runs on various AWS services, including Amazon EC2 and Amazon DynamoDB.
How can the developer determine the length of time of the application's calls to the various downstream AWS services?
A. Enable VPC Flow Logs and analyze them in Amazon OpenSearch Service.
B. Use Amazon CloudWatch Logs to analyze application logs for the various calls.
C. Enable detailed monitoring for the EC2 instances in Amazon CloudWatch.
D. Implement AWS X-Ray with client handlers for the various downstream calls.
Answer: D (LEAVE A REPLY)
To measure how long an application spends calling downstream AWS services (for example, DynamoDB, S3, SNS, etc.) in a distributed system, the most effective AWS-native approach is AWS X-Ray with appropriate instrumentation. X-Ray provides distributed tracing by capturing end-to-end request paths and breaking each request into segments and subsegments with precise timing information. When the application is instrumented with the X-Ray SDK, client handlers/interceptors automatically create subsegments for calls to AWS services, recording latency, response status, and error/throttle indicators.
This directly satisfies the requirement: determine the length of time of calls to various downstream services.
In the X-Ray trace map and trace details, the developer can see per-service timing, identify which dependency is slow, and distinguish application processing time from downstream call time. This is specifically designed for performance bottleneck analysis and root cause investigation across distributed components.
Option A (VPC Flow Logs) captures network flow metadata (source/destination, ports, bytes, accept/reject), not application-level request timing to AWS service APIs, and it won't show the latency breakdown per downstream service call.
Option B can work only if the application is already logging detailed timing for each call, but this is manual, inconsistent across services, and does not give a unified distributed trace view. It is also higher effort than using standard tracing instrumentation.
Option C increases CloudWatch metric granularity for EC2 but does not show per-request downstream service call durations.
Therefore, implementing AWS X-Ray with client handlers is the correct solution.

NEW QUESTION: 237
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: (SHOW ANSWER)
Lambda Destinations on Failure: Allow routing asynchronous function invocations to specified resources (like another Lambda function) upon failure.
Error Handling: The error-handling Lambda receives details about the failure, enabling logging and custom actions.
Direct Integration: This solution leverages native Lambda functionality for a simpler implementation.

NEW QUESTION: 238
A company notices that credentials that the company uses to connect to an external software as a service (SaaS) vendor are stored in a configuration file as plaintext.
The developer needs to secure the API credentials and enforce automatic credentials rotation on a quarterly basis.
Which solution will meet these requirements MOST securely?
A. Use AWS Key Management Service (AWS KMS) to encrypt the configuration file. Decrypt the configuration file when users make API calls to the SaaS vendor. Enable rotation.
B. Retrieve temporary credentials from AWS Security Token Service (AWS STS) every 15 minutes. Use the temporary credentials when users make API calls to the SaaS vendor.
C. Store the credentials in AWS Secrets Manager and enable rotation. Configure the API to have Secrets Manager access.

D. Store the credentials in AWS Systems Manager Parameter Store and enable rotation. Retrieve the credentials when users make API calls to the SaaS vendor.

Answer: (SHOW ANSWER)

Store the credentials in AWS Secrets Manager and enable rotation. Configure the API to have Secrets Manager access. This is correct. This solution will meet the requirements most securely, because it uses a service that is designed to store and manage secrets such as API credentials. AWS Secrets Manager helps you protect access to your applications, services, and IT resources by enabling you to rotate, manage, and retrieve secrets throughout their lifecycle1. You can store secrets such as passwords, database strings, API keys, and license codes as encrypted values2. You can also configure automatic rotation of your secrets on a schedule that you specify3. You can use the AWS SDK or CLI to retrieve secrets from Secrets Manager when you need them4. This way, you can avoid storing credentials in plaintext files or hardcoding them in your code.

NEW QUESTION: 239

A company is building a serverless application that uses AWS Lambda functions. The company needs to create a set of test events to test Lambda functions in a development environment. The test events will be created once and then will be used by all the developers in an 1AM developer group. The test events must be editable by any of the 1AM users in the 1AM developer group.

Which solution will meet these requirements?

A. Create and store the test events in Amazon S3 as JSON objects. Allow S3 bucket access to all 1AM users.

B. Create and store the test events in Amazon DynamoDB. Allow access to DynamoDB by using 1AM roles.

C. Create the test events. Configure the event sharing settings to make the test events shareable.

D. Create the test events. Configure the event sharing settings to make the test events private.

Answer: C (LEAVE A REPLY)

NEW QUESTION: 240

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

B. The createDeploymer.t method must be called so the API can be redeployed 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 updateAuthorizer method must be called to update the API's authorizer to include the newly created API key
Answer: A (LEAVE A REPLY)

NEW QUESTION: 241
A developer is building a three-tier web application that should be able to handle a minimum of 5000 requests per minute. Requirements state that the web tier should be completely stateless while the application maintains session state for the users.
How can session data be externalized, keeping latency at the LOWEST possible value?
A. Create an Amazon RDS instance, then implement session handling at the application level to leverage a database inside the RDS database instance for session data storage.
B. Implement a shared file system solution across the underlying Amazon EC2 instances, then implement session handling at the application level to leverage the shared file system for session data storage.
C. Create an Amazon ElastiCache (Memcached) cluster, then implement session handling at the application level to leverage the cluster for session data storage.
D. Create an Amazon DynamoDB table, then implement session handling at the application level to leverage the table for session data storage.
Answer: C (LEAVE A REPLY)
* Why Option C is Correct:
* Amazon ElastiCache (Memcached) provides low-latency, in-memory caching suitable for session storage. It ensures stateless web tier operations and supports the high throughput of 5000 requests per minute.
* Why Other Options are Incorrect:
* Option A:RDS has higher latency compared to in-memory caching solutions like ElastiCache.
* Option B:Shared file systems introduce additional complexity and are not ideal for low-latency session data storage.
* Option D:DynamoDB has low latency but is less performant than ElastiCache for in-memory session management.
* AWS Documentation References:
* Amazon ElastiCache for Session State Management

NEW QUESTION: 242
A developer is automating a new application deployment with AWS SAM. The new application has one AWS Lambda function and one Amazon S3 bucket. The Lambda function must access the S3 bucket to only read objects.
How should the developer configure AWS SAM to grant the necessary read permission to the S3 bucket?
A. Reference a second Lambda authorizer function.
B. Add a custom S3 bucket policy to the Lambda function.
C. Create an Amazon SQS topic for only S3 object reads. Reference the topic in the template.
D. Add the S3ReadPolicy template to the Lambda function's execution role.
Answer: D (LEAVE A REPLY)
Step-by-Step Breakdown:
Requirement Summary:
Use AWS SAM to deploy:
1 Lambda Function
1 S3 Bucket
Lambda needs read-only access to the S3 bucket
Solution must be expressed via AWS SAM template
Option A: Reference a second Lambda authorizer function
Incorrect: Lambda authorizers are used in API Gateway for authentication, not for granting S3 permissions.
Option B: Add a custom S3 bucket policy to the Lambda function
Incorrect: Bucket policies control who can access the bucket, not what the Lambda function can do.
The permission must be granted to the Lambda's IAM execution role.
Option C: Create an Amazon SQS topic for only S3 object reads
Incorrect: SQS cannot read objects from S3 and is not relevant to this scenario.
Option D: Add the S3ReadPolicy template to the Lambda function's execution role Correct: AWS SAM provides managed policy templates like AmazonS3ReadOnlyAccess and shortcuts like S3ReadPolicy.
You can apply these to the Lambda's execution role using the Policies: section in your SAM template.
Example SAM YAML:
yaml
CopyEdit

Resources:
MyFunction:
Type: AWS::Serverless::Function
Properties:
CodeUri: my-code/
Handler: app.handler
Runtime: python3.11
Policies:
- S3ReadPolicy:
BucketName: !Ref MyBucket
MyBucket:
Type: AWS::S3::Bucket
Properties:
BucketName: my-personal-bucket-name
SAM Policy Templates: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide
/serverless-policy-templates.html
Example using S3ReadPolicy: https://docs.aws.amazon.com/serverless-application-model/latest
/developerguide/serverless-policy-templates.html#s3-readpolicy

NEW QUESTION: 243
A developer wants to add request validation to a production environment Amazon API Gateway API. The developer needs to test the changes before the API is deployed to the production environment. For the test, the developer will send test requests to the API through a testing tool.
Which solution will meet these requirements with the LEAST operational overhead?
A. Export the existing API to an OpenAPI file. Create a new API. Import the OpenAPI file. Modify the new API to add request validation. Perform the tests. Modify the existing API to add request validation.
Deploy the existing API to production.
B. Modify the existing API to add request validation. Deploy the updated API to a new API Gateway stage. Perform the tests. Deploy the updated API to the API Gateway production stage.
C. Create a new API. Add the necessary resources and methods, including new request validation. Perform the tests. Modify the existing API to add request validation. Deploy the existing API to production.
D. Clone the existing API. Modify the new API to add request validation. Perform the tests. Modify the existing API to add request validation. Deploy the existing API to production.
Answer: (SHOW ANSWER)

Amazon API Gateway allows you to create, deploy, and manage a RESTful API to expose backend HTTP endpoints, AWS Lambda functions, or other AWS services1. You can use API Gateway to perform basic validation of an API request before proceeding with the integration request1. When the validation fails, API Gateway immediately fails the request, returns a 400 error response to the caller, and publishes the validation results in CloudWatch Logs1.
To test changes before deploying to a production environment, you can modify the existing API to add request validation and deploy the updated API to a new API Gateway stage1. This allows you to perform tests without affecting the production environment. Once testing is complete and successful, you can then deploy the updated API to the API Gateway production stage1.
This approach has the least operational overhead as it avoids unnecessary creation of new APIs or exporting and importing of APIs. It leverages the existing infrastructure and only requires changes in the configuration of the existing API1.

NEW QUESTION: 244
A developer is creating a new REST API by using Amazon API Gateway and AWS Lambda. The development team tests the API and validates responses for the known use cases before deploying the API to the production environment.
The developer wants to make the REST API available for testing by using API Gateway locally.
Which AWS Serverless Application Model Command Line Interface (AWS SAM CLI) subcommand will meet these requirements?
A. Sam local invoke
B. Sam local generate-event
C. Sam local start-lambda
D. Sam local start-api
Answer: (SHOW ANSWER)
The AWS Serverless Application Model Command Line Interface (AWS SAM CLI) is a command-line tool for local development and testing of Serverless applications2. The sam local start-api subcommand of AWS SAM CLI is used to simulate a REST API by starting a new local endpoint3. Therefore, option D is correct.

NEW QUESTION: 245
A developer is writing an application that will retrieve sensitive data from a third-party system. The application will format the data into a PDF file. The PDF file could be more than 1 MB. The application will encrypt the data to disk by using AWS Key Management Service (AWS KMS). The application will decrypt the file when a user requests to download it. The retrieval and formatting portions of the application are complete.

The developer needs to use the GenerateDataKey API to encrypt the PDF file so that the PDF file can be decrypted later. The developer needs to use an AWS KMS symmetric customer managed key for encryption.
Which solutions will meet these requirements?
A. Write the encrypted key from the GenerateDataKey API to disk for later use. Use the plaintext key from the GenerateDataKey API and a symmetric encryption algorithm to encrypt the file.
B. Write the plain text key from the GenerateDataKey API to disk for later use. Use the encrypted key from the GenerateDataKey API and a symmetric encryption algorithm to encrypt the file.
C. Write the encrypted key from the GenerateDataKey API to disk for later use. Use the plaintext key from the GenerateDataKey API to encrypt the file by using the KMS Encrypt API
D. Write the plain text key from the GenerateDataKey API to disk for later use. Use the encrypted key from the GenerateDataKey API to encrypt the file by using the KMS Encrypt API
Answer: A (LEAVE A REPLY)
The GenerateDataKey API returns a data key that is encrypted under a symmetric encryption KMS key that you specify, and a plaintext copy of the same data key1. The data key is a random byte string that can be used with any standard encryption algorithm, such as AES or SM42. The plaintext data key can be used to encrypt or decrypt data outside of AWS KMS, while the encrypted data key can be stored with the encrypted data and later decrypted by AWS KMS1.
In this scenario, the developer needs to use the GenerateDataKey API to encrypt the PDF file so that it can be decrypted later. The developer also needs to use an AWS KMS symmetric customer managed key for encryption. To achieve this, the developer can follow these steps:
Call the GenerateDataKey API with the symmetric customer managed key ID and the desired length or specification of the data key. The API will return an encrypted data key and a plaintext data key.
Write the encrypted data key to disk for later use. This will allow the developer to decrypt the data key and the PDF file later by using AWS KMS.
Use the plaintext data key and a symmetric encryption algorithm to encrypt the PDF file. The developer can use any standard encryption library or tool to perform this operation, such as OpenSSL or AWS Encryption SDK.
Discard the plaintext data key from memory as soon as possible after using it. This will prevent unauthorized access or leakage of the data key.

NEW QUESTION: 246

A company has a large amount of data in an Amazon DynamoDB table. A large batch of data is appended to the table once each day. The company wants a solution that will make all the existing and future data in DynamoDB available for analytics on a long-term basis. Which solution meets these requirements with the LEAST operational overhead?

A. Configure DynamoDB incremental exports to Amazon S3.

B. Configure Amazon DynamoDB Streams to write records to Amazon S3.

C. Configure Amazon EMR to copy DynamoDB data to Amazon S3.

D. Configure Amazon EMR to copy DynamoDB data to Hadoop Distributed File System (HDFS).

Answer: A (LEAVE A REPLY)

* Why Option A is Correct:DynamoDB supports incremental exports to Amazon S3 natively, making data analytics-ready with minimal operational overhead.

* Why Other Options are Incorrect:

* Option B: DynamoDB Streams require additional processing logic to write to S3, increasing complexity.

* Option C & D: Using EMR for data movement adds unnecessary operational overhead compared to native exports.

* AWS Documentation References:

* DynamoDB Exports to S3

NEW QUESTION: 247

A company has three AWS Lambda functions written in Node.js. The Lambda functions include a mix of custom code and open source modules. When bugs are occasionally detected in the open source modules, all three Lambda functions must be patched. What is the MOST operationally efficient solution to deploy a patched open source library for all three Lambda functions?

A. Create a custom AWS CloudFormation public registry extension. Reference a GitHub repository that hosts the open source modules in the extension. Configure CloudFormation to scan the repository once each day. Write an AWS SAM template to redeploy the three Lambda functions upon a scan notification change.

B. Create an Amazon CloudFront distribution with an Amazon S3 bucket as the origin. Upload the patched modules to Amazon S3 when needed. Modify each Lambda function to download the patched modules from the CloudFront distribution during cold starts.

C. Launch an Amazon EC2 instance. Host a private open source module registry on the EC2 instance.
Upload the modified open source modules to the private registry when needed. Modify each Lambda function deployment script to download the modules from the private registry. Redeploy the three Lambda functions.

D. Create a Lambda layer with the open source modules. Modify all three Lambda functions to use the layer. Remove the open source modules from each Lambda function.

Patch the Lambda layer with the modified open source modules when needed. Update the Lambda functions to reference the new layer version.

Answer: (SHOW ANSWER)

Comprehensive and Detailed 250 to 300 words of Explanation (AWS-doc-aligned):

The most operationally efficient way to share and patch common dependencies across multiple Lambda functions is to use AWS Lambda layers. A Lambda layer is an independently managed deployment artifact that can contain libraries and other dependencies. Multiple functions can reference the same layer, which avoids packaging identical open source modules into each function bundle.

When a vulnerability or bug is found in a shared module, you patch it once by publishing a new layer version that contains the updated dependencies. Then you update each Lambda function configuration to reference the new layer version. This centralizes dependency management, reduces duplicated artifacts, and makes rollouts and rollbacks cleaner (you can switch layer versions quickly). It also keeps function deployment packages smaller, which often improves deployment speed and can reduce cold-start overhead associated with larger bundles.

The other options introduce unnecessary operational burden:

* Downloading modules at cold start (CloudFront/S3) increases latency and adds failure modes at runtime.
* Hosting a private registry on EC2 requires patching, scaling, availability management, and security hardening-high operational overhead.
* A CloudFormation registry extension and daily scans are not a standard or efficient mechanism for packaging runtime Node.js dependencies into Lambda functions.

Lambda layers are a documented AWS-native feature intended for precisely this use case: sharing common code and libraries across functions while enabling controlled versioning. Therefore, creating a layer for the open source modules and updating the functions to consume the layer is the most efficient and maintainable approach.

NEW QUESTION: 248

A developer is building a highly secure healthcare application using serverless components. This application requires writing temporary data to /Imp storage on an AWS Lambda function.

How should the developer encrypt this data?

A. Use an on-premises hardware security module (HSM) to generate keys, where the Lambda function requests a data key from the HSM and uses that to encrypt data on all requests to the function.

B. Set up the Lambda function with a role and key policy to access an AWS KMS key. Use the key to generate a data key used to encrypt all data prior to writing to Amp storage.

C. Enable Amazon EBS volume encryption with an AWS KMS key in the Lambda function configuration so that all storage attached to the Lambda function is encrypted.

D. Use OpenSSL to generate a symmetric encryption key on Lambda startup. Use this key to encrypt the data prior to writing to /tmp.
Answer: (SHOW ANSWER)

NEW QUESTION: 249
An application routinely processes a large number of Amazon S3 GET requests each second. A developer wants to increase the number of requests that the application can handle in parallel.
What should the developer do to achieve this goal?
A. Configure AWS Global Accelerator for Amazon S3.
B. Move all Amazon S3 objects into a single object prefix.
C. Partition Amazon S3 objects by object prefixes.
D. Configure AWS Direct Connect for Amazon S3.
Answer: C (LEAVE A REPLY)
Amazon S3 automatically scales to handle high request rates, but request parallelism is optimized when objects are distributed across multiple prefixes. AWS documentation explains that S3 internally partitions data by key name, and distributing objects across multiple prefixes allows requests to be spread across multiple partitions.
When many GET requests target objects within the same prefix, they may contend for the same internal partition, limiting throughput. By designing object keys with multiple prefixes-such as including hashed or randomized components-applications can significantly increase parallel request handling.
Options A and D do not address S3 request partitioning. Global Accelerator improves network latency for supported endpoints but does not increase S3 request parallelism. Direct Connect improves network consistency but does not change how S3 scales requests internally. Option B worsens the problem by concentrating traffic into a single prefix.
Therefore, partitioning objects by prefixes is the correct and AWS-recommended solution.

NEW QUESTION: 250
An ecommerce application is running behind an Application Load Balancer. A developer observes some unexpected load on the application during non-peak hours. The developer wants to analyze patterns for the client IP addresses that use the application. Which HTTP header should the developer use for this analysis?
A. The X-Forwarded-Proto header
B. The X-F Forwarded-Host header
C. The X-Forwarded-For header
D. The X-Forwarded-Port header
Answer: (SHOW ANSWER)
The HTTP header that the developer should use for this analysis is the X-Forwarded-For header. This header contains the IP address of the client that made the request to the

Application Load Balancer. The developer can use this header to analyze patterns for the client IP addresses that use the application. The other headers either contain information about the protocol, host, or port of the request, which are not relevant for the analysis.
Reference: How Application Load Balancer works with your applications

NEW QUESTION: 251
A developer is migrating a containerized application from an on-premises environment to an Amazon ECS cluster.
In the on-premises environment, the container uses a Docker file to store the application. Service dependency configurations such as databases, caches, and storage volumes are stored in a docker-compose.yml file.
Both files are located at the top level of the code base that the developer needs to containerize. When the developer deploys the code to Amazon ECS, the instructions from the Docker file are carried out. However, none of the configurations from docker-compose.yml are applied.
The developer needs to resolve the error and ensure the configurations are applied.
A. Store the file path for the docker-compose.yml file as a Docker label. Add the label to the ECS cluster's container details.
B. Add the details from the docker-compose.yml file to an ECS task definition. Associate the task with the ECS cluster.
C. Create a namespace in the ECS cluster. Associate the docker-compose.yml file to the namespace.
D. Update the service type of the ECS cluster to REPLICA, and redeploy the stack.
Answer: B (LEAVE A REPLY)
* Why Option B is Correct:Amazon ECS does not natively process docker-compose.yml files. Instead, the configurations from docker-compose.yml must be converted into ECS-compatible configurations within a task definition. Task definitions are the primary way to specify container configurations in ECS, including service dependencies like databases, caches, and volumes.
* Steps to Resolve the Error:
* Extract the configurations from the docker-compose.yml file.
* Map the dependencies and settings to the appropriate ECS task definition fields.
* Deploy the task definition to the ECS cluster.
* Why Other Options are Incorrect:
* Option A:Docker labels do not directly impact ECS task execution or integrate with ECS service configurations.
* Option C:ECS namespaces do not exist as a feature.
* Option D:Changing the service type to REPLICA does not resolve the issue of missing service dependency configurations.
* AWS Documentation References:
* Amazon ECS Task Definitions

* Migrating Docker Compose Workloads to ECS