

SAP-C02 Demo

Topic 1, Exam Pool A

Question: 1

A company is providing weather data over a REST-based API to several customers. The API is hosted by Amazon API Gateway and is integrated with different AWS Lambda functions for each API operation. The company uses Amazon Route 53 for DNS and has created a resource record of weather.example.com. The company stores data for the API in Amazon DynamoDB tables. The company needs a solution that will give the API the ability to fail over to a different AWS Region. Which solution will meet these requirements?

- A. Deploy a new set of Lambda functions in a new Region. Update the API Gateway API to use an edge-optimized API endpoint with Lambda functions from both Regions as targets. Convert the DynamoDB tables to global tables.
- B. Deploy a new API Gateway API and Lambda functions in another Region. Change the Route 53 DNS record to a multivalue answer. Add both API Gateway APIs to the answer. Enable target health monitoring. Convert the DynamoDB tables to global tables.
- C. Deploy a new API Gateway API and Lambda functions in another Region. Change the Route 53 DNS record to a failover record. Enable target health monitoring. Convert the DynamoDB tables to global tables.
- D. Deploy a new API Gateway API in a new Region. Change the Lambda functions to global functions. Change the Route 53 DNS record to a multivalue answer. Add both API Gateway APIs to the answer. Enable target health monitoring. Convert the DynamoDB tables to global tables.

Answer: C

Explanation:

<https://docs.aws.amazon.com/apigateway/latest/developerguide/dns-failover.html>

Question: 2

A company is running a two-tier web-based application in an on-premises data center. The application layer consists of a single server running a stateful application. The application connects to a PostgreSQL database running on a separate server. The application's user base is expected to grow significantly, so the company is migrating the application and database to AWS. The solution will use Amazon Aurora PostgreSQL, Amazon EC2 Auto Scaling, and Elastic Load Balancing. Which solution will provide a consistent user experience that will allow the application and database tiers to scale?

- A. Enable Aurora Auto Scaling for Aurora Replicas. Use a Network Load Balancer with the least outstanding requests routing algorithm and sticky sessions enabled.
- B. Enable Aurora Auto Scaling for Aurora writers. Use an Application Load Balancer with the round robin routing algorithm and sticky sessions enabled.
- C. Enable Aurora Auto Scaling for Aurora Replicas. Use an Application Load Balancer with the round robin routing and sticky sessions enabled.
- D. Enable Aurora Scaling for Aurora writers. Use a Network Load Balancer with the least outstanding requests routing algorithm and sticky sessions enabled.

Answer: C

Explanation:

Aurora Auto Scaling enables your Aurora DB cluster to handle sudden increases in connectivity or workload. When the connectivity or workload decreases, Aurora Auto Scaling removes unnecessary Aurora Replicas so that you don't pay for unused provisioned DB instances

Question: 3

A company uses a service to collect metadata from applications that the company hosts on premises. Consumer devices such as TVs and internet radios access the applications. Many older devices do not support certain HTTP headers and exhibit errors when these headers are present in responses. The company has configured an on-premises load balancer to remove the unsupported headers from responses sent to older devices, which the company identified by the User-Agent headers.

The company wants to migrate the service to AWS, adopt serverless technologies, and retain the ability to support the older devices. The company has already migrated the applications into a set of AWS Lambda functions.

Which solution will meet these requirements?

- A. Create an Amazon CloudFront distribution for the metadata service. Create an Application Load Balancer (ALB). Configure the CloudFront distribution to forward requests to the ALB. Configure the ALB to invoke the correct Lambda function for each type of request. Create a CloudFront function to remove the problematic headers based on the value of the User-Agent header.
- B. Create an Amazon API Gateway REST API for the metadata service. Configure API Gateway to invoke the correct Lambda function for each type of request. Modify the default gateway responses to remove the problematic headers based on the value of the User-Agent header.
- C. Create an Amazon API Gateway HTTP API for the metadata service. Configure API Gateway to invoke the correct Lambda function for each type of request. Create a response mapping template to remove the problematic headers based on the value of the User-Agent. Associate the response data mapping with the HTTP API.
- D. Create an Amazon CloudFront distribution for the metadata service. Create an Application Load Balancer (ALB). Configure the CloudFront distribution to forward requests to the ALB. Configure the ALB to invoke the correct Lambda function for each type of request. Create a Lambda@Edge function that will remove the problematic headers in response to viewer requests based on the value of the User-Agent header.

Answer: D

Explanation:

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-examples.html>

Question: 4

A company is running a traditional web application on Amazon EC2 instances. The company needs to refactor the application as microservices that run on containers. Separate versions of the application exist in two distinct environments: production and testing. Load for the application is variable, but the minimum load and the maximum load are known. A solutions architect needs to design the updated application with a serverless architecture that minimizes operational complexity.

Which solution will meet these requirements MOST cost-effectively?

- A. Upload the container images to AWS Lambda as functions. Configure a concurrency limit for the associated Lambda functions to handle the expected peak load. Configure two separate Lambda integrations within Amazon API Gateway: one for production and one for testing.
- B. Upload the container images to Amazon Elastic Container Registry (Amazon ECR). Configure two auto scaled Amazon Elastic Container Service (Amazon ECS) clusters with the Fargate launch type to handle the expected load. Deploy tasks from the ECR images. Configure two separate Application Load Balancers to direct traffic to the ECS clusters.
- C. Upload the container images to Amazon Elastic Container Registry (Amazon ECR). Configure two auto scaled Amazon Elastic Kubernetes Service (Amazon EKS) clusters with the Fargate launch type to handle the expected load. Deploy tasks from the ECR images. Configure two separate Application Load Balancers to direct traffic to the EKS clusters.
- D. Upload the container images to AWS Elastic Beanstalk. In Elastic Beanstalk, create separate environments and deployments for production and testing. Configure two separate Application Load Balancers to direct traffic to the Elastic Beanstalk deployments.

Answer: D

Explanation:

minimizes operational + microservices that run on containers = AWS Elastic Beanstalk

Question: 5

A company has a multi-tier web application that runs on a fleet of Amazon EC2 instances behind an Application Load Balancer (ALB). The instances are in an Auto Scaling group. The ALB and the Auto Scaling group are replicated in a backup AWS Region. The minimum value and the maximum value for the Auto Scaling group are set to zero. An Amazon RDS Multi-AZ DB instance stores the application's data.

a. The DB instance has a read replica in the backup Region. The application presents an endpoint to end users by using an Amazon Route 53 record.

The company needs to reduce its RTO to less than 15 minutes by giving the application the ability to automatically fail over to the backup Region. The company does not have a large enough budget for an active-active strategy.

What should a solutions architect recommend to meet these requirements?

- A. Reconfigure the application's Route 53 record with a latency-based routing policy that load balances traffic between the two ALBs. Create an AWS Lambda function in the backup Region to promote the read replica and modify the Auto Scaling group values. Create an Amazon CloudWatch alarm that is based on the HTTPCode_Target_5XX_Count metric for the ALB in the primary Region. Configure the CloudWatch alarm to invoke the Lambda function.
- B. Create an AWS Lambda function in the backup Region to promote the read replica and modify the Auto Scaling group values. Configure Route 53 with a health check that monitors the web application and sends an Amazon Simple Notification Service (Amazon SNS) notification to the Lambda function when the health check status is unhealthy. Update the application's Route 53 record with a failover policy that routes traffic to the ALB in the backup Region when a health check failure occurs.
- C. Configure the Auto Scaling group in the backup Region to have the same values as the Auto Scaling group in the primary Region. Reconfigure the application's Route 53 record with a latency-based routing policy that load balances traffic between the two ALBs. Remove the read replica. Replace the read replica with a standalone RDS DB instance. Configure Cross-Region Replication between the RDS DB instances by using snapshots and Amazon S3.
- D. Configure an endpoint in AWS Global Accelerator with the two ALBs as equal weighted targets. Create an AWS Lambda function in the backup Region to promote the read replica and modify the Auto Scaling group values. Create an Amazon CloudWatch alarm that is based on the HTTPCode_Target_5XX_Count metric for the ALB in the primary Region. Configure the CloudWatch alarm to invoke the Lambda function.

Answer: B

Explanation:

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html>