AWS CHEAT SHEETS: <https://tutorialsdojo.com/aws-cheat-sheets/>

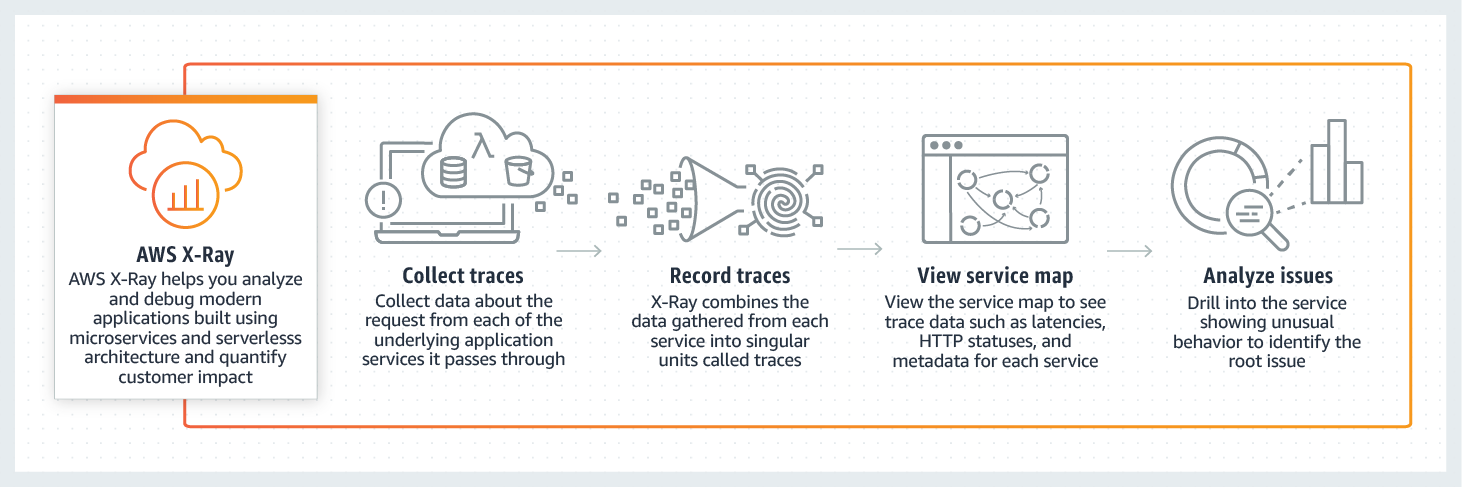
**Question 1: Incorrect**

You have created a Node.js Lambda function that updates a DynamoDB table and sends an email notification via Amazon SNS. However, upon testing, the function is not working as expected. Which of the following is the BEST way to troubleshoot this issue?

* Use AWS CloudTrail
* Use AWS X-Ray(Correct)
* Use Amazon Inspector
* Use Amazon CloudWatch(Incorrect)

#### 

AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application’s underlying components.



You can use X-Ray to analyze both applications in development and in production, from simple three-tier applications to complex microservices applications consisting of thousands of services.

AWS X-Ray works with Amazon EC2, Amazon EC2 Container Service (Amazon ECS), AWS Lambda, and AWS Elastic Beanstalk. You can use X-Ray with applications written in Java, Node.js, and .NET that are deployed on these services.

**Amazon CloudWatch is incorrect** because although you can troubleshoot the issue by checking the logs, it is still better to use AWS X-Ray as it enables you to analyze and debug your serverless application more effectively.

Amazon Inspector is incorrect because this is primarily used for EC2 and not for Lambda.

AWS CloudTrail is incorrect because this will only enable you to track all API calls to your Lambda, DynamoDB, and SNS. It is still better to use AWS X-Ray to debug your application.

Reference:

<https://aws.amazon.com/xray/>

<https://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-logs.html>

Check out these AWS X-Ray and Lambda Cheat Sheets:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-x-ray/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/>

AWS XRAY - Cheat sheet

* PURPOSE:
  + Analyzes and debugs production, distributed applications
* IDENTIFY
  + Performance bottlenecks
  + Edge case errors
  + Hard to detect issues
* CONCEPTS
  + Segment
    - Provides the name of resources
    - Details about the request
    - Details about the work done
  + Subsegments
    - Contains additional details about a call to an aws service
      * Inferred segments and downstream nodes
        + X-Ray creates segments if resources do not generate their own segments
    - Represent your application's view of a downstream call as a client.
  + Service Graph
    - X-Ray utilizes data that the applications sent to generate a service graph
    - Each AWS resource that sends data to AWS appears as a service in the graph
    - A JSON document that contains information of the application and resources. Information is retained for 30 days
  + Edges
    - Connect services that work together to serve requests
    - Connect clients to your application
    - Application to downstream services
  + Trace ID
    - Tracks the path of a request through the application
    - Collects all the segments generated by a single request
      * Typically a HTTP request
  + Sampling Algorithm
    - By default, the X-Ray SDK records the first request each second, and five percent of any additional requests
    - You can look at individual requests or use filter expressions to find traces related to specific paths or users
      * Groups
        + A collection of traces defined by a filter expression
      * Annotations
        + Simple key value pairs that are indexed for use with filter expressions.
        + Used to record data that you want to use to group traces

A segment can contain multiple annotations

System-Defined Annotations

Data added to segments about AWS services

User-Defined Annotations

Metadata added to a segment by a developer

* + - * Metadata
        + Key-value pairs with values of any type, that are not indexed
        + Use metadata to record data you want to store and trace
    - When an exception, error or fault occurs, the X-Ray SDK will record details about the error, including the stack trace, if available
* FEATURES
  + Use with applications running on: just integrate the x-ray SDK w/ app and install x-ray agent
    - EC2
    - ECS
    - Lambda
    - Elastic Beanstalk
  + Provides end-to-end, cross-service,application centric view of requests by aggregating the data into a single unit called a trace
  + Captures metadata for requests to
    - MySQL
    - PostgreSQL
    - DynamoDB
    - SQS
    - SNS
  + Trace sampling rate can be set to whatever suits your applications best
  + Creates a mpa of services used by the application using the aggregated trace data.
  + Add annotations to data emitted from specific points in the application
* HOW IT WORKS
  + Data from services as **segments**
  + **Groups segments** that have common requests into **traces**
  + X-Ray processes the **traces** to generate a **service graph**
* X-Ray SDK
  + - Interceptors add via code to trace incoming HTTP requests
    - Client handlers to instrument AWS SDK clients that your application uses to call other AWS services
    - HTTP client to use to instrument calls to other internal and external HTTP web services
  + AWS X-Ray supports tracing for applications that are written in Node.js, Java, and .NET
  + The SDK sends JSON segment documents to an X-Ray daemon for process listening for UDP traffic
  + The X-Ray daemon buffers segments in a queue and uploads them to X-Ray in batches
* AWS Service Integration and Service Graph
  + Service integration can include adding tracing headers to incoming requests, sending trace data to X-Ray, or runnin the X-Ray daemon
  + You can use the service graph to identify bottlenecks, latency spikes, and other issues
  + Four types of X-Ray integration
    - Active Instrumentation - samples and instruments incominging requests
    - Passive instrumentation – instruments requests that have been sampled by another service
    - Request tracing– adds a tracing header to all incoming requests and propagates it downstream
    - Tooling – runs the x ray daemon to receive segments from the X-ray sdk
  + The following services provide X-Ray integration
    - Lambda
      * Active
      * Passive
      * Adds two nodes to the service map one for lambda service and one for the function
    - API Gateway
      * Active
      * Passive
      * Adds a node for the gateway stage
    - Elastic Load Balancing
      * Request tracing on applicaiton load balancers
      * Adds trace ID to request headers before sending it to a target group
      * AWS Elastic Beanstalk – tooling
* PRICING
  + Pay based on the number of traces recorded, retrieved, and scanned
  + Max size of trace is 500KB
  + Trace data retained 30 days no extra cost
* INSTRUMENTING AWS X\_RAY
  + <https://tutorialsdojo.com/instrumenting-your-application-with-aws-x-ray/>

AWS LAMBDA - Cheat sheet

* OVERVIEW
  + Serverless compute service
  + Execute code only when needed and scales automatically
  + Stateless functions and have no affinity to the underlying infrastructure
  + Choose the amount of memory and lambda will allocate proportional
    - CPU
    - Network bandwidth
    - Disk Input/Output
  + SOC, HIPAA, PCI, ISO compliant
  + Supports
    - Node.js
    - Java
    - C#
    - Go
    - Python
    - Ruby
* COMPONENTS
  + Functions
    - A script or program that runs in lambda passes invocation events to your function.
    - The function processes an even and return a response
  + Runtimes
    - Lambda runtimes allow functions in different languages to run in the same base execution environment.
    - The runtime sits in-between the Lambda service and function code relaying invocation events, context information, and responses between the function and runtime
  + Layers
    - Distribution mechanisms for libraries, custom runtimes, and other function dependencies
    - Manage your in-development function code independently from the unchanging code and resources that it uses
  + Event Source
    - An AWS service or a custom service that triggers the function and executes its logic
  + Downstream Resources
    - An AWS service called on a lambda trigger
  + Log Streams
    - Automatically monitors function invocations and reports metrics to cloudwatch
    - Annotate function code with custom logging statements that allow for analysis of the execution flow and performance
  + AWS Serverless Application Model
* FEATURES
  + Lambda stores code in S3 and encrypts it at rest
  + Package your code and dependencies in a deployment package

**Question 2: Incorrect**

An internal web application is hosted in a custom VPC with multiple private subnets only. Every EC2 instance that will be provisioned on this VPC will require access to an S3 bucket to pull configuration files as well as to push application logs.

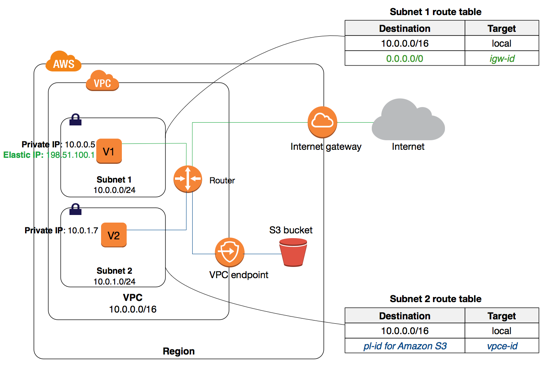
Which of the following options is the most suitable solution to use in this scenario?​

* Create a VPC endpoint for S3.(Correct)
* Use the AWS SDK for your application and issue the aws configure CLI command to store your access keys, which will be referred to by the SDK.​
* Create an IAM Role and attach it to each EC2 instance.(Incorrect)
* Store the IAM user and password in the application code to access the S3 bucket.

#### 

In this scenario, the key point that you have to understand is that S3 is not part of your VPC, unlike your EC2 instances, EBS volumes, ELBs, and other services which typically reside within your private network. An EC2 instance needs to have access to the Internet, via the Internet Gateway or a NAT Instance/Gateway in order to access S3. Alternatively, you can also create a VPC endpoint so your private subnet would be able to connect to S3.

To visualize this, take a look at the diagram below:



Hence, creating a VPC endpoint for S3 is the correct answer.

The option that says "Use the AWS SDK for your application and issue the aws configure CLI command to store your access keys, which will be referred to by the SDK." is incorrect because this set up will only work if the EC2 instance is in a public subnet or if there is a VPC endpoint to your S3 bucket. Moreover, it is recommended to use IAM Role instead of storing the access keys to your EC2 instances.

The option that says "Create an IAM Role and attach it to each EC2 instance." is incorrect because although this is the recommended way to grant access to S3, there would still be no connectivity between your instance and S3. Your VPC should have an Internet Gateway, a NAT Instance/Gateway or a VPC endpoint in order to establish a connection between these two services.

The option that says "Store the IAM user and password in the application code to access the S3 bucket" is incorrect because this would be a security risk since the credentials will be on plain text. Moreover, this does not resolve the connectivity issue between the two services.

References:

<https://aws.amazon.com/blogs/aws/new-vpc-endpoint-for-amazon-s3>

<https://aws.amazon.com/premiumsupport/knowledge-center/connect-s3-vpc-endpoint/>

Check out this Amazon VPC Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-vpc/>

AWS VPC

* + Create a virtual network in the cloud dedicated to your AWS account where you can launch AWS resources
  + Amazon VPC is the networking layer of amazon EC2
  + A VPC spans all the AZ in the region.
  + Upon a VPC creation you can add one or more subnets in each AZ
* KEY CONCEPTS
  + A VPC allows one to specify an IP address range for the VPC
  + Allocated subnets
  + SUBNET
    - A range of IP addresses in one’s VPC
    - YOu can launch AWS resources into a specified subnet
    - TWO TYPES
      * PUBLIC SUBNET
        + For resources that must be connected to the internet
      * PRIVATE SUBNET
        + For resources that will not be connected to the internet
  + Associate security groups
    - To protect the AWS resources in each subnet use security groups and network access control lists (ACL)
  + Configure route tables
  + Expand your VPC by adding secondary IP ranges
* EC2-VPC vs EC2-CLASSIC (basically what EC2-VPC has that EC2-CLASSIC does not)
  + Assign static private IPv4 addresses to your instances that persist across starts and stops
  + Optionally associate an Ipv6 CIDR block to your VPC and assign IPv6 addresses to your instances
  + Assign multiple IP addresses to your instances
  + Define network interfaces, and attach one or more network interfaces to your instances
  + Change security group membership of your instances while they’re running
  + Control the outbound traffic from your instances (egress filtering) in addition to controlling the inbound traffic to them (ingress filtering)
  + Add an additional layer of access control to your instances in the form of network access control lists (ACL)
  + Run you instances on a single tenant hardware
* Default vs NonDefault VPC
* DEFAULT
  + If your account supports the EC2-VPC platform only , ti comes with a default VPC that has a default subnet in each AZ
  + Your default VPC includes an internet gateway, which allows your instances to communicate with the internet and each default subnet is a public subnet
  + Each instance that you launch into a default subnet has a private IPv3 address and a public IPv4 address
  + To allow an instance in your VPC to initiate outbound connections to the internet but prevent unsolicited inbound connections from the internet, you can use a network address translation (NAT) advice for IPv4 traffic
  + You can optionally associate an Amazon-provided IPv6 CIDR block with your VPC and assign IPv6 addresses to your instances. IPv6 traffic is separate from IPv4 traffic; your route tables must include separate routes for IPv6 traffic
* NON-DEFAULT VPC
  + You can create your own non-default VPC, and configure it as you need. Subnets that you create in your

VPC SUBNETS

* Upon creation one must specify a range of IPv4 addresses for the VPC in the form of Classless Inter-Domain Routing (CIDR) block
  + This is the PRIMARY CIDR BLOCK
* You can add one or more subnets in each AZ of the VPC’s region
* You specify the CIDR block for a subnet, which is a subset the VPC primary CIDR BLOCK
* A CIDR block must not overlap with any existing CIDR block that's associated with the VPC
* TYPES:
  + Public Subnet - has an internet gateway
  + Private Subnet - Does not have an internet gateway
  + VPN-only Subnet - Has a virtual private gateway instead
* IPv4 CIDR block size should be between a /16 netmask (65,536 IP addresses) and /28 netmask (16 IP addresses)
* The **first four** IP addresses and the **last four** IP address in each subnet CIDR block are **not available** for you to use and cannot assigned to an instance
* You cannot increase or decrease the size of an existing CIDR block
* When you associate CIDR block with your VPC, a route is automatically added to your VPC route tables to enable routing within the VPC (the destination is th eCIDR block and the target is local)
* There is a limit on the number of CIDR blocks you can associate with a VPC and the number of routes you can add to a route table
* RULES to apply when you add IPv4 CIDR blocks to a VPC that’s pare of a VPC peering connection

Question 3: Incorrect

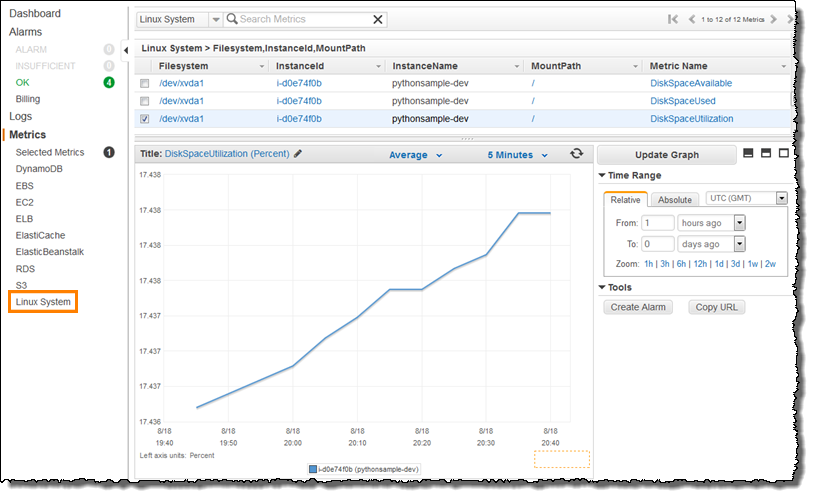
A mission-critical application is required to have a monitoring system which can provide immediate insight into its sub-minute activity. You are required to collect the data of all of the users who are currently logged in to the system every 10 seconds.

Which of the following options is the MOST suitable solution that you should do to meet the above requirements?

* ​Publish a high-resolution custom metric to CloudWatch.
* (Correct)
* Enable detailed monitoring.(Incorrect)
* Enable enhanced monitoring.
* ​Publish a custom metric to CloudWatch using the PutMetricData API with the --storage-resolution parameter set to its default value.

#### 

You can publish your own metrics to CloudWatch using the AWS CLI or an API. You can view statistical graphs of your published metrics with the AWS Management Console. CloudWatch stores data about a metric as a series of data points. Each data point has an associated time stamp. You can even publish an aggregated set of data points called a statistic set.



Each metric is one of the following:

- Standard resolution, with data having a one-minute granularity

- High resolution, with data at a granularity of one second

Metrics produced by AWS services are standard resolution by default. When you publish a custom metric, you can define it as either standard resolution or high resolution. When you publish a high-resolution metric, CloudWatch stores it with a resolution of 1 second, and you can read and retrieve it with a period of 1 second, 5 seconds, 10 seconds, 30 seconds, or any multiple of 60 seconds.

High-resolution metrics can give you more immediate insight into your application's sub-minute activity. Keep in mind that every PutMetricData call for a custom metric is charged, so calling PutMetricData more often on a high-resolution metric can lead to higher charges.

Therefore, the correct answer in this scenario is to publish a high-resolution custom metric to CloudWatch.

The option that says: "Publish a custom metric to CloudWatch using the PutMetricData API with the --storage-resolution parameter set to its default value" is incorrect because the default value of the --storage-resolution parameter is 60, which stores data in one-minute granularity. This parameter should be set to 1 in order to configure it as a high-resolution metric.

The option that says: "Enable detailed monitoring" is incorrect because it will only store metric data in one-minute granularity. Moreover, you have to publish a custom metric for this scenario since the type of data that you have to monitor is specific only to your application.

The option that says: "Enable enhanced monitoring" is incorrect because this type of monitoring is primarily used only in RDS.

References:

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html>

<https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-custom-metrics/>

<https://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-data.html>

Check out this Amazon CloudWatch Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/>

Tutorials Dojo's AWS Certified Developer Associate Exam Study Guide:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-developer-associate/>

**Question 4: Incorrect**

A developer is building a web application which requires a multithreaded event-based key/value cache store that will cache result sets from database calls. You need to run large nodes with multiple cores for your cache layer and it should scale up or down as the demand on your system increases and decreases.

Which of the following is the MOST suitable service that you should use?

* Amazon CloudFront
* Amazon ElastiCache for Memcached(Correct)
* AWS Greengrass
* Amazon ElastiCache for Redis(Incorrect)

#### 

Redis and Memcached are popular, open-source, in-memory data stores. Although they are both easy to use and offer high performance, there are important differences to consider when choosing an engine. Memcached is designed for simplicity while Redis offers a rich set of features that make it effective for a wide range of use cases.

In this scenario, Redis can provide a much more durable and powerful cache layer to the prototype distributed system, however, you should take note of one keyword in the requirement: multithreaded. In terms of commands execution, Redis is mostly a single-threaded server. It is not designed to benefit from multiple CPU cores unlike Memcached, however, you can launch several Redis instances to scale out on several cores if needed.

Memcached is a more suitable choice since the scenario specifies that the system will run large nodes with multiple cores or threads which Memcached can adequately provide.

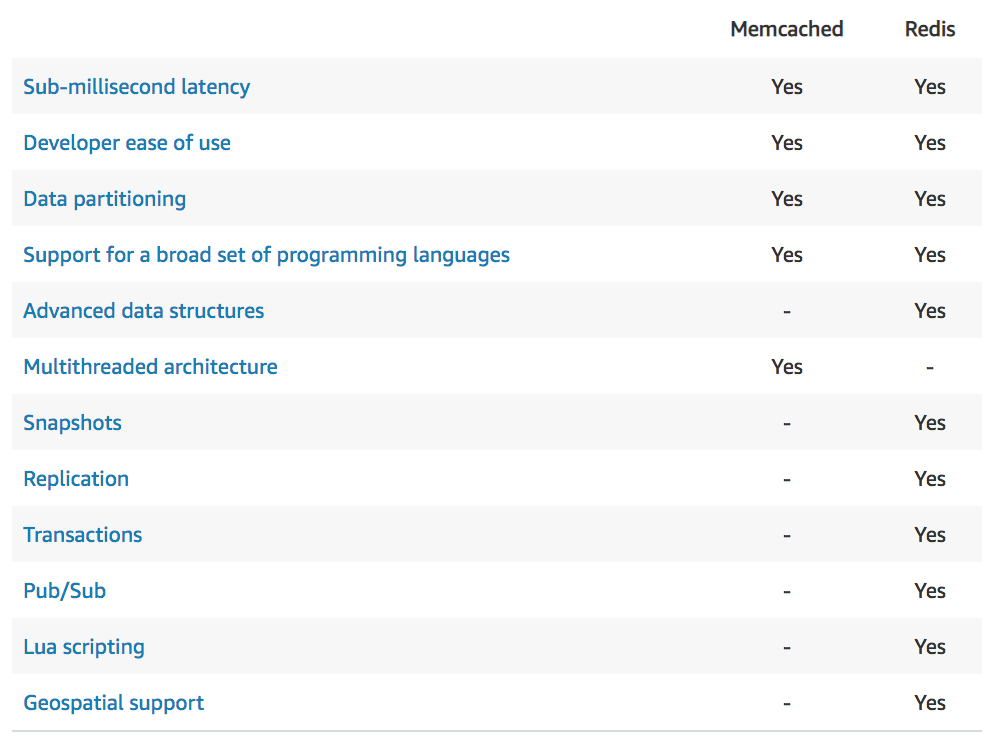
You can choose Memcached over Redis if you have the following requirements:

- You need the simplest model possible.

- You need to run large nodes with multiple cores or threads.

- You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases.

- You need to cache objects, such as a database.



This is why the most suitable answer to this scenario is Amazon ElastiCache for Memcached.

Amazon ElastiCache for Redis is incorrect because it does not totally support a multithreaded architecture, unlike Memcached. Although Redis has more features compared with Memcached, the scenario requires that the cache layer is multithreaded. This is why Memcached is a more suitable cache engine to choose from instead of Redis.

Amazon CloudFront is incorrect because it is primarily used as a Content Delivery Network (CDN) service which delivers your entire website, including dynamic, static, streaming, and interactive content using a global network of edge locations.

AWS IoT Greengrass is incorrect because this service is primarily used to enable connected devices to run AWS Lambda functions, execute predictions based on machine learning models, keep device data in sync, and communicate with other devices securely even without an Internet connection. Hence, this is not a suitable option for this scenario.

References:

<https://aws.amazon.com/elasticache/redis-vs-memcached>

<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/SelectEngine.html>

<https://aws.amazon.com/caching/aws-caching/>

Redis (cluster mode enabled vs disabled) vs Memcached:

<https://tutorialsdojo.com/aws-cheat-sheet-redis-cluster-mode-enabled-vs-disabled-vs-memcached/>

Comparison of AWS Services Cheat Sheets:

<https://tutorialsdojo.com/comparison-of-aws-services-for-udemy-students/>

LITERALLY the only thing that memcache has over redis is the multithreading capacity

You can choose Memcached over Redis if you have the following requirements:

- You need the simplest model possible.

- You need to run large nodes with multiple cores or threads.

- You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases.

- You need to cache objects, such as a database.

**Question 5: Incorrect**

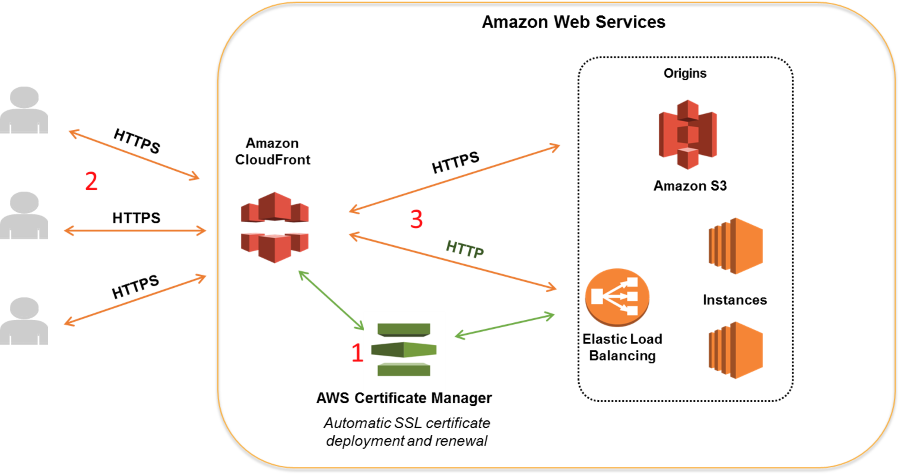
A website is hosted in an Auto Scaling group of EC2 instances behind an Application Load Balancer. It also uses CloudFront with a default domain name to distribute its static assets and dynamic contents. However, the website has a poor search ranking as it doesn't use a secure HTTPS/SSL on its site.

Which are the possible solutions that the developer can implement in order to set up HTTPS communication between the viewers and CloudFront? (Select TWO)

* Use a self-signed SSL/TLS certificate in the ALB which is stored in a private S3 bucket.
* (Incorrect)
* Set the Viewer Protocol Policy to use HTTPS Only.
* (Correct)​
* Set the Viewer Protocol Policy to use Redirect HTTP to HTTPS.(Correct)
* Configure the ALB to use its default SSL/TLS certificate.
* ​Use a self-signed certificate in the ALB.

You can configure one or more cache behaviors in your CloudFront distribution to require HTTPS for communication between viewers and CloudFront. You also can configure one or more cache behaviors to allow both HTTP and HTTPS, so that CloudFront requires HTTPS for some objects but not for others.

If you're using the domain name that CloudFront assigned to your distribution, such as dtut0ria1sd0jo.cloudfront.net, you can change the Viewer Protocol Policy setting for one or more cache behaviors to require HTTPS communication by setting it as either Redirect HTTP to HTTPS, or HTTPS Only.



If your origin is an Elastic Load Balancing load balancer, you can use a certificate provided by AWS Certificate Manager (ACM). You can also use a certificate that is signed by a trusted third-party certificate authority and imported into ACM. Note that you can't use a self-signed certificate for HTTPS communication between CloudFront and your origin.

Hence, setting the Viewer Protocol Policy to use Redirect HTTP to HTTPS and setting the Viewer Protocol Policy to use HTTPS Only are the correct answers in this scenario.

Using a self-signed SSL/TLS certificate in the ALB which is stored in a private S3 bucket is incorrect because you don't need to add an SSL certificate if you only require HTTPS for communication between the viewers and CloudFront. You should only do this if you require HTTPS between your origin and CloudFront. In addition, you can't use a self-signed certificate in this scenario even though it is stored in a private S3 bucket. You need to use either a certificate from ACM or a third-party certificate.

Configuring the ALB to use its default SSL/TLS certificate is incorrect because there is no default SSL certificate in ELB, unlike what we have in CloudFront.

Using a self-signed certificate in the ALB is incorrect because adding an SSL certificate in the ELB is not required. Moreover, you can't use a self-signed certificate in this scenario.

References:

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-cloudfront-to-custom-origin.html#using-https-cloudfront-to-origin-certificate>

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-viewers-to-cloudfront.html>

Check out this Amazon CloudFront Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/>

AMAZON CLOUDFRONT

* A web service that speeds up distribution of your static and dynamic web content to you users.
* A Content Delivery Network (CDN service)
* Delivers content through a worldwid network of data centers calle **edge locations**.
  + Users requests content that you’re serving with cloudfront,
    - IF the content is already in teh edge location with the lowest latency cloudfront delivers it immediately
    - IF NOT in the edge location Cloud Fron retrieves it from the origin that you’ve defined

HOW CLOUDFRONT DE$LIVERS CONTENT

* One specifies the  **Origin Servers**  examples are an S3 bucket, or an HTTP server.
* Cloud Formation gets the files which will then be distributed from CloudFront edge locations around the world.
* Upload your files to origin servers. These files are referred to as **objects**
* Create a  **Cloudfront distribution,** this tells a cloudfront which origin servers to get objects from when a user request the object through you website or application.
  + At the same time you specify details for CloudFront to log in requests and if you want distribution to be enabled as soon as it’s created
* CloudFront assigns a domain name to the new distribution
* CloudFront sends your distribution configuration to all of its **edge locations**

SUPPORTS

* CloudFront supports the WebSocket Protocol as well as the HTTP protocol with the following HTTP methods:
  + Get
  + Head
  + Post
  + Put
  + Delete
  + Options
  + Patch
* Wildcard CNAME

WEBSOCKETS:

* A computer communications protocol, providing full-duplex communication channels over a single TCP connection
* Distinct from HTTP. Both protocols work at the 7 layer of the OSI model and depends on TCP at layer 4.
* The websocket enables interactions between a web browser or other client application and a webserver with lower overhead than the half-duplex alternatives such as HTTP polling.

CLOUDFRONT REGIONAL EDGE CACHE

* That brings content closer to your viewers, even when the content is not popular enough to stay in a edge location

CLOUDFRONT ORIGINS

* S3 bucket
  + You place any objects that you want cloudfront to deliver in an s3 bucket
* S3 buckets configured as website endpoints for your origin
* Mediastore container or a mediapackage channel
  + You can set up an S3 bucket that is configured as a mediastore container, or create channel and endpoints with media package. Then you create and configure a distribution in cloudfront to stream the video
* EC2 or other custom origins
  + A custom origin is an HTTP server like a web server
* CloudFront origin group for origin failover
  + Use origin failover to designate a primary origin for CloudFrontplus a second ORigin that CloudFront automatically switches to when the primary origin returns a specific HTTP status code failure responses

HOW IT WORKS:

* Objects are cached for 24 hours by default. However you can remove file from edge caches before they expire
* You can configure cloudfront to automatically compress files of certain types and serve compressed files when viewers request include the Accept-Encoding: gzip in requestheader
* CloudFront can cache different versions of your content based on the values of query string parameters

CLOUDFRONT DISTRIBUTIONS

White papers <https://d0.awsstatic.com/whitepapers/Building%20Static%20Websites%20on%20AWS.pdf>

* Create a cloudfront distribution to tell cloudfront where you want content to be delivered from, and the details about how to track and manage content delivery.
* You create a distribution and choose the configuration settings you want:
  + **Your content origin**–that is , the amazon S3 bucket, mediaPackage channel, or HTTP server from which CloudFront gets the files to distribute. You can specify any combination of up to 25 S3 buckets, channels, and/ or HTTP servers as your origins.
  + **Access -** whether you wna te files to be available to everyone or restrict access to some users.
  + **Security** - whether you want CloudFFront to require users to use HTTPs to access your content,
  + **Cooke or query string forwarding** - whether you want cloudfront to forward cookies or query strings to your origin
  + **Geo-restrictions -** whether you want CloudFront to prevent users in selected countries from accessing your content.
  + **Access Logs -** whether you want cloud front to create access logs that show viewer activity
* You can use distributions to serve the following types content over HTTP or HTTPS:
  + CONTENT TYPES
    - Static and dynamic download content
    - Video on demand in different formats, such as APple HTTP Live Streaming (HLS) and Microsoft Smooth Streaming
    - Live events such as a meeting conference or concert in real time
* Values that you specify when you create or update a distribution
  + **Delivery Method**
    - Web or RTMP (Real Time Messaging Protocol)
  + **Origin Settings** 
    - Information About one or more locations where you store the original versions of your web content
  + **Cache Behavior Settings** 
    - Lets you configure a variety of CLoudFront functionality for a given URL path patter for files on your website
  + **Custom Error Pages and Error Caching**
  + **Restrictions** 
    - If you need to prevent users in selected countries from accessing your content, you can configure your Cloudfront distribution either to allow users ina whitelist of specified countries to acces s your content or to not allow users ina blacklist of specified countries to access your content

CACHE BEHAVIOR SETTINGS

* The functionality that you can configure for each cache behaviour includes:
  + The path pattern
  + If you have configured multiple origins for your CloudFront Distribution, which origin you want CloudFron to forward your requests to
  + Whether to forward query strings to your origin
  + Whether accessing the specified files requires signed URLS
  + Whether to require users to use HTTPS to access those files
  + The minimum amount of time that those files stay in the CloudFront cache regardless of the value of any cache-control headers that your origin added to the files

PRICE CLASS

* Choose the price class that corresponds with the maximum price that you want to pay for CloudFront service. By default CloudFront servers your objects from edge locations in all CloudFront regions.

**Question 6: Incorrect**

You are developing a Lambda function which processes event notifications from Amazon S3. It is expected that the function will have:

- 50 requests per second

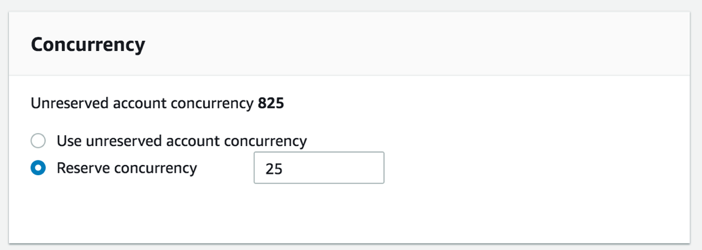
- 100 seconds to complete each request

What should you do to prevent any issues when the function has been deployed and becomes operational?

* Request for AWS to increase the limit of your concurrent executions.(Correct)​
* No additional action needed since Lambda will automatically scale based on the incoming requests.(Incorrect)
* Implement exponential backoff in your application.
* ​Increase the concurrency limit of the function.

#### 

Concurrent executions refers to the number of executions of your function code that are happening at any given time. You can estimate the concurrent execution count, but the concurrent execution count will differ depending on whether or not your Lambda function is processing events from a poll-based event source.



If you create a Lambda function to process events from event sources that aren't poll-based (for example, Lambda can process every event from other sources, like Amazon S3 or API Gateway), each published event is a unit of work, in parallel, up to your account limits. Therefore, the number of invocations these event sources make influences the concurrency.

You can use this formula to estimate the capacity used by your function:

**concurrent executions = (invocations per second) x (average execution duration in seconds)**

For example, consider a Lambda function that processes Amazon S3 events. Suppose that the Lambda function takes on average three seconds and Amazon S3 publishes 10 events per second. Then, you will have 30 concurrent executions of your Lambda function. See the calculation shown below to visualize the process:

= (10 events per second) x (3 seconds average execution duration)

= 30 concurrent executions

In this scenario, it is expected that the Lambda function takes an average of 100 seconds for every execution with 50 requests per second. Using the formula above, the function will have 5,000 concurrent executions.

= (50 events per second) x (100 seconds average execution duration)

= 5,000 concurrent executions

AWS Lambda dynamically scales function execution in response to increased traffic, up to your [concurrency limit](https://docs.aws.amazon.com/lambda/latest/dg/limits.html). Under sustained load, your function's concurrency bursts to an initial level between 500 and 3000 concurrent executions that varies per region. After the initial burst, the function's capacity increases by an additional 500 concurrent executions each minute until either the load is accommodated, or the total concurrency of all functions in the region hits the limit.

**By default, AWS Lambda limits the total concurrent executions across all functions within a given region to 1000.** This limit can be raised by requesting for AWS to increase the limit of the concurrent executions of your account.

Since the expected concurrent executions of the Lambda function will exceed the default concurrency limit, the best thing to do here is to request for AWS to increase the limit of your concurrent executions.

Choosing to do no additional action since Lambda will automatically scale based on the incoming requests is incorrect because the dynamic scaling of AWS Lambda has its limits. Because the value of the expected concurrency executions has exceeded the default limit, it is best to contact AWS to increase the concurrent executions of your account to prevent any throttling issues when the function has been deployed and becomes operational.

Implementing an exponential backoff in your application is incorrect because this doesn't address the concurrency issue of your Lambda function. This will just configure your application to have progressively longer waits between API call retries for consecutive error responses.

Increasing the concurrency limit of the function is incorrect because, by default, you can only set the limit as high as 900 per function, which is quite insufficient to handle the expected 5,000 concurrency executions. To properly provide the required capacity needed by the function, you have to request for AWS to increase the concurrency limit of your account.

References:

<https://docs.aws.amazon.com/lambda/latest/dg/running-lambda-code.html>

<https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html#function-configuration>

Check out this AWS Lambda Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/>

DYNAMO NOTE

- [Global secondary index](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html) — an index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions.

- [Local secondary index](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/LSI.html) — an index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.

**Question 11: Incorrect**

The users of a social media website must be authenticated using social identity providers such as Twitter, Facebook, and Google. Users can login to the site which will allow them to upload their selfies, memes, and other media files in an S3 bucket. As an additional feature, you should also enable guest user access to certain sections of the website.

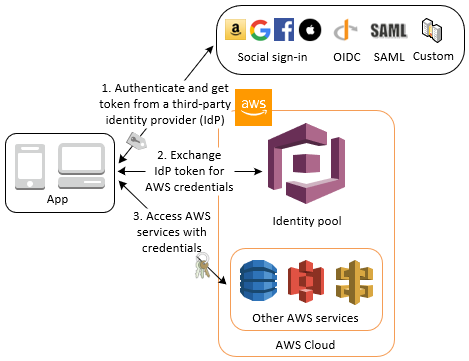
Which of the following should you do to accomplish this task?

* Create a User Pool in Amazon Cognito and enable access to unauthenticated identities.(Incorrect)
* Create a custom identity broker which integrates with the AWS Security Token Service and supports unauthenticated access.
* Create an Identity Pool in Amazon Cognito and enable access to unauthenticated identities.(Correct)
* ​Integrate AWS Single Sign-On with your website.

#### 

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, or Google.

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.



Amazon Cognito identity pools (federated identities) support user authentication through Amazon Cognito user pools, federated identity providers—including Amazon, Facebook, Google, and SAML identity providers—as well as unauthenticated identities. This feature also supports [Developer Authenticated Identities (Identity Pools)](https://docs.aws.amazon.com/cognito/latest/developerguide/developer-authenticated-identities.html), which lets you register and authenticate users via your own back-end authentication process.

Hence, creating an Identity Pool in Amazon Cognito and enabling access to unauthenticated identities is the most suitable answer for this scenario.

Creating a User Pool in Amazon Cognito with unauthenticated identities enabled is incorrect because you should have created an Identity Pool instead. Take note that a User Pool doesn't have an option to enable unauthenticated identities. Moreover, you won't be able to provide your users access to upload their media files to S3 using a User Pool.

Creating a custom identity broker which integrates with the AWS Security Token Service and supports unauthenticated access is incorrect because this is not a suitable solution in this scenario. You only need to build a custom identity broker application if your identity store is not compatible with SAML 2.0 which is required for identity federation.

Integrating AWS Single Sign-On with your website is incorrect because this is only used to help you manage access and permissions to custom applications that support Security Assertion Markup Language (SAML) 2.0 and commonly used third-party software as a service (SaaS) applications. This is primarily used for existing corporate identities and not for social identity providers.

References:

<https://aws.amazon.com/premiumsupport/knowledge-center/cognito-user-pools-identity-pools/>

<https://docs.aws.amazon.com/cognito/latest/developerguide/getting-started-with-identity-pools.html>

AMAZON COGNITO

* A user management and authentication service that can be integrated to your web or mobile applications. Amazon Cognito also enables you to authenticate users through an external identity provider and provides temporary security credentials to access your app's bracken resources in AWS or any service bethind Amazon API Gateway. Amazon Cognito works with external identity providers that support SAML or Openid connect, social identity providers you and you can also integrate your own identity provider
* An amazon cognito ID token is represented as JSON Web Token (JWT) Amazon gonitoe uses son web tokens for token authentication

USER POOLS

* User pools are user dirctories that provide sign-up and sign-in options for your app users
* Users can sign in to your we or mobile app through Amaszon Cognito, or federate through a third-party identity provider ( IdP).
* You can use the aliasing feature to enable your users to sign up or sign in with an email address and password or a phone number and a password
* User pools are each created in one AWS region and they store teh user profile data only in that region you can also send user data to a different AWS region
* Manage USers
  + After you create a user pool, you can create, confirm and manage users accounts.
  + Amazon cognito users pools groups lets you manage your users and their access to resources by mapping IAM roles to groups
  + User accounts are added to your user pool in one of the following ways:
    - The user signs up in your user pool clients app, which can be a mobile or web app.
    - You can import the user’s account into your user pool
    - You can create the user’s account in your user pool and invite the user to sign in
    - Sign up authflow

IDENTITY POOLS

* Use this feature if you want to federate users to your aws services
* Identity pools enable you to grant your users temporary aws credentials to access aws services such as S3 DyDB
* Identity pools support anonymous guest users, as well s the following identity providers:
  + Amazon Cognito user pools
  + Social Sign In with facebook, google, and login with amazon
  + Openid connect (OIDC) providers
  + SAML identity providers
  + Developer authenticated identities
* To save user profile information, your identity pool needs to be integrated with a user ool
* Amazon cognito identity pools can support unauthenticated identities by providing a unique identifier and AWS credentials forusers who do not authenticate witha n identity provider.
* The permissions fore ach authenticated an dnon-authenticated user are controlled through IAM roles that you create
* Once you have an OpenID connect token you can then trade this for temp AWS credntials via the ASSUMEORLEWITHWEBIDENTITY API call in AWS security Token Service (STS). This call is no different whan if you were using FB Google) or login with amazon directly except thtyou are passing an Amazon cognito token instead of a token from one of the other public providers

DEPLOYMENT

#### 

In ElasticBeanstalk, you can choose from a variety of deployment methods:

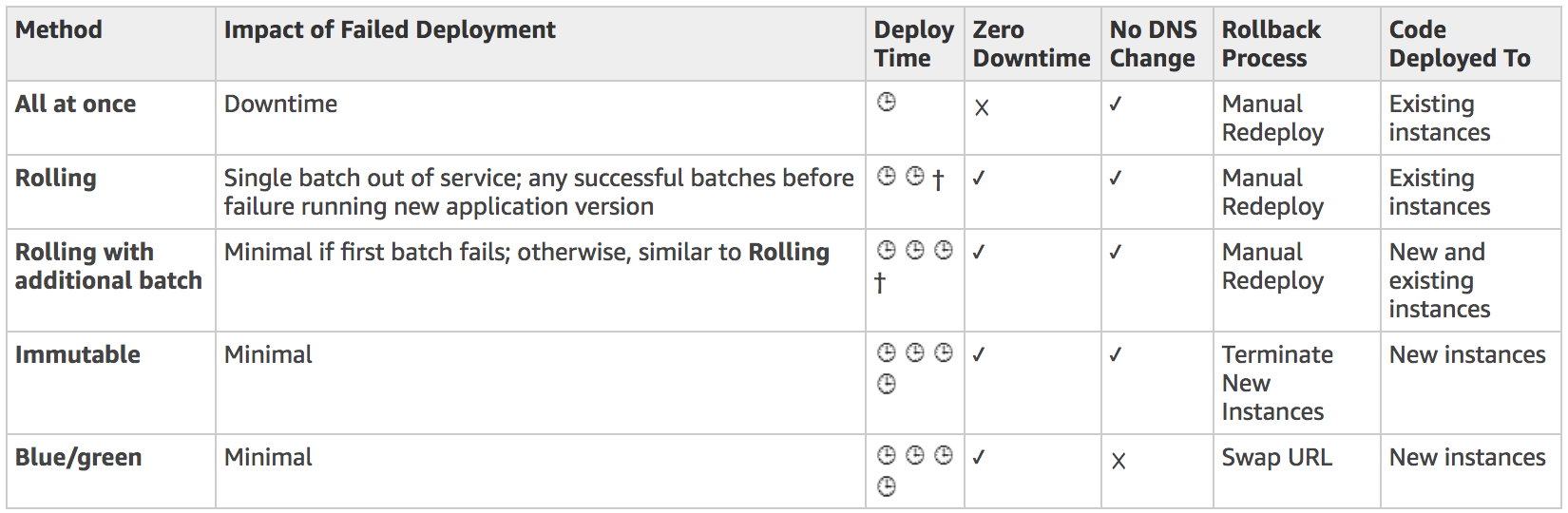
- **All at once** – Deploy the new version to all instances simultaneously. All instances in your environment are out of service for a short time while the deployment occurs.

- **Rolling** – Deploy the new version in batches. Each batch is taken out of service during the deployment phase, reducing your environment's capacity by the number of instances in a batch.

- **Rolling with additional batc**h – Deploy the new version in batches, but first launch a new batch of instances to ensure full capacity during the deployment process.

- **Immutable** – Deploy the new version to a fresh group of instances by performing an [immutable update](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environmentmgmt-updates-immutable.html).

- **Blue/Green** - Deploy the new version to a separate environment, and then swap CNAMEs of the two environments to redirect traffic to the new version instantly.



**Question 13: Incorrect**

A developer has recently completed a new version of a serverless application that is ready to be deployed using AWS SAM. There is a requirement that the traffic should shift from the previous Lambda function to the new version gradually, in the shortest time possible.

Which deployment configuration is the MOST suitable one to use in this scenario?

* CodeDeployDefault.HalfAtATime
* CodeDeployDefault.LambdaLinear10PercentEvery2Minutes
* CodeDeployDefault.LambdaCanary10Percent5Minute(Correct)
* CodeDeployDefault.LambdaLinear10PercentEvery1Minute(Incorrect)

#### 

If you use AWS SAM to create your serverless application, it comes built-in with CodeDeploy to help ensure safe Lambda deployments. There are various deployment preference types that you can choose from.

For example:

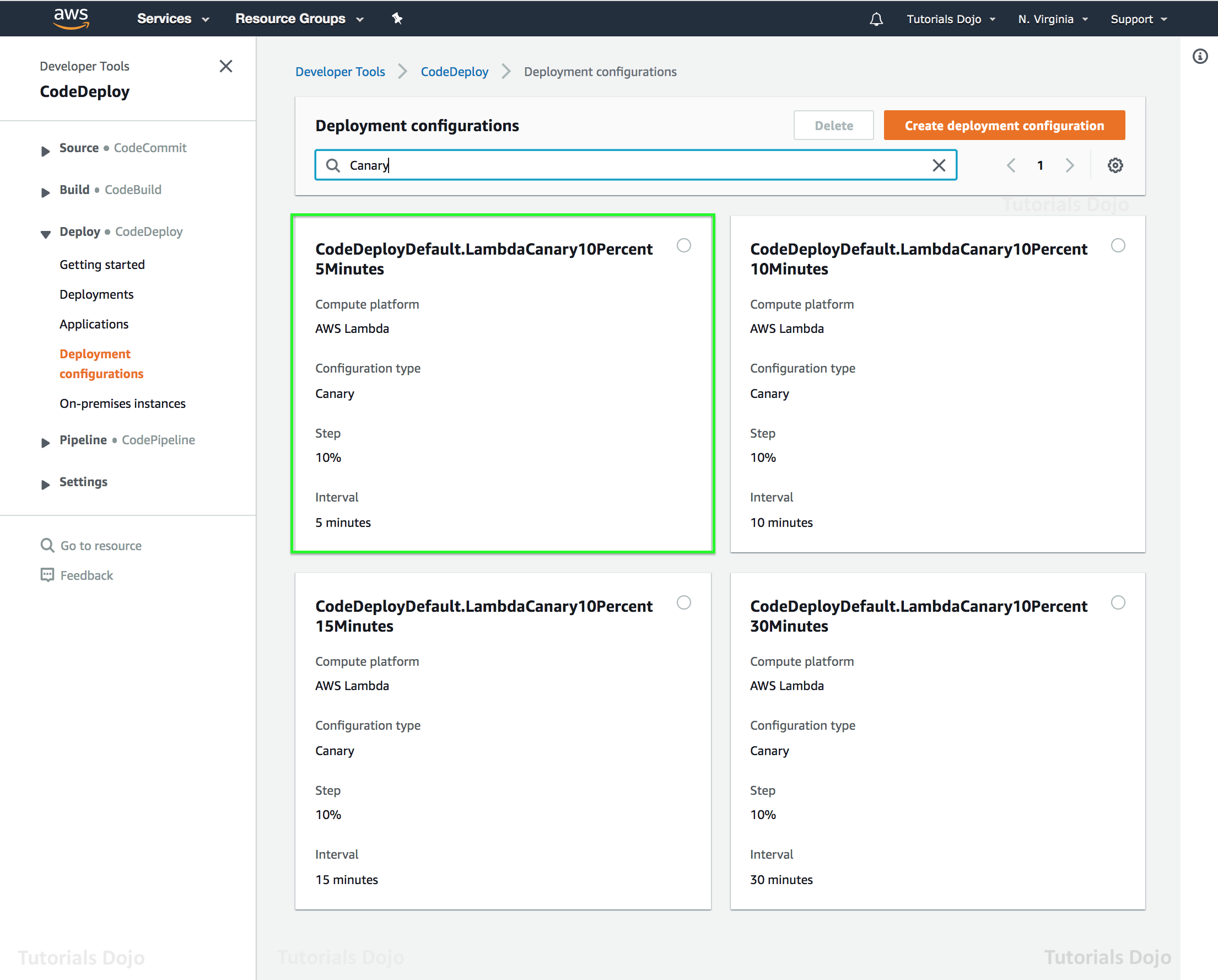
If you choose Canary10Percent10Minutes then 10 percent of your customer traffic is immediately shifted to your new version. After 10 minutes, all traffic is shifted to the new version.

However, if your pre-hook/post-hook tests fail, or if a CloudWatch alarm is triggered, CodeDeploy rolls back your deployment. The following table outlines other traffic-shifting options that are available:

- Canary: Traffic is shifted in two increments. You can choose from predefined canary options. The options specify the percentage of traffic that's shifted to your updated Lambda function version in the first increment, and the interval, in minutes, before the remaining traffic is shifted in the second increment.

- Linear: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic that's shifted in each increment and the number of minutes between each increment.

- All-at-once: All traffic is shifted from the original Lambda function to the updated Lambda function version at once.



Hence, the CodeDeployDefault.LambdaCanary10Percent5Minutes option is correct because 10 percent of your customer traffic is immediately shifted to your new version. After 5 minutes, all traffic is shifted to the new version. This means that the entire deployment time will only take 5 minutes

CodeDeployDefault.HalfAtATime is incorrect because this is only applicable for EC2/On-premises compute platform and not for Lambda.

CodeDeployDefault.LambdaLinear10PercentEvery1Minute is incorrect because it will add 10 percent of the traffic linearly to the new version every minute. Hence, all traffic will be shifted to the new version only after 10 minutes

CodeDeployDefault.LambdaLinear10PercentEvery2Minutes is incorrect because it will add 10 percent of the traffic linearly to the new version every 2 minutes. Hence, all traffic will be shifted to the new version only after 20 minutes.

References:

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/automating-updates-to-serverless-apps.html>

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-deploying.html>

**Question 15: Incorrect**

An API gateway with a Lambda proxy integration takes a long time to complete its processing. There were also occurrences where some requests timed out. You want to monitor the responsiveness of your API calls as well as the underlying Lambda function.

Which of the following CloudWatch metrics should you use to troubleshoot this issue? (Select TWO)

​

* Latency(Correct)
* Count
* CacheMissCount(Incorrect)
* CacheHitCount
* IntegrationLatency(Correct)

#### 

You can monitor API execution using CloudWatch, which collects and processes raw data from API Gateway into readable, near real-time metrics. These statistics are recorded for a period of two weeks, so that you can access historical information and gain a better perspective on how your web application or service is performing. By default, API Gateway metric data is automatically sent to CloudWatch in one-minute periods.

The metrics reported by API Gateway provide information that you can analyze in different ways. The list below shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list.

- Monitor the IntegrationLatency metrics to measure the responsiveness of the backend.

- Monitor the Latency metrics to measure the overall responsiveness of your API calls.

- Monitor the CacheHitCount and CacheMissCount metrics to optimize cache capacities to achieve a desired performance.

Hence, the correct metrics that you have to use in this scenario are Latency and IntegrationLatency.

Count is incorrect because this metric simply gets the total number of API requests in a given period.

CacheMissCount is incorrect because this metric just gets the number of requests served from the backend in a given period when API caching is enabled. The Sum statistic represents this metric, namely, the total count of the cache misses in the given period.

CacheHitCount is incorrect because this fetches the number of requests served from the API cache in a given period. The Sum statistic represents this metric, namely, the total count of the cache hits in the given period.

References:

<https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-metrics-and-dimensions.html>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/monitoring-cloudwatch.html>

Check out this Amazon API Gateway Cheat Sheet:

[https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway](https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway/)

AMAZON API GATEWAY

* Enables developers to create, publish, maintain, monitor, and secure APIs at any scale.
* HIPAA eligible
* Allows creating deployin, and managing a restful api to expose backed http endpoints, lambda functins or other aws resources
* Together with lambda, api gateway forms the app facing part of the aws serverless infrastructure

CONCEPTS

* API deployment
  + A point in time snapshot of you api gateway api resources and methods to be available for clients to use the deployment must be associated with one or more API stages
* API endpoints
  + Host names APIs in API Gateway, which are deployed to a specific region and of the format rest-api-id.execute-api.region.amazonaws.com
* API key
  + An alphanumeric string that API Gateway uses to identify an app developer who uses your API
* API Stage
  + A logical reference to a lifecycle state of your API. API stages are identified by API ID andstage name.
* Model
  + Data schema specifying the data structure of a request or response payload
* Private API
  + An ApI that is exposed through interface VPC endpoints and isolated from the public internet
* Private integration
  + An API gateway integration type for a client to access resources inside a customer’s VPC through a private API endpoint without exposing the resources to the public internet
* Proxy integration
  + You can set up a proxy integration as an HTTP proxy integration type or a Lambda proxy integration type
    - For the HTTP proxy integration, API gateway passes the entire request and response between the frontend and an HTTP backend
    - For the lambda proxy integration, API Gateway sends the entire request as an input to a backend Lambda function
* Usage Plan
  + Provides selected api clients with access to one or more deployed APIs. YOu can use a usage plan to configure throttling and quota limits, which are enfored on individual client api keys.

API ENDPOINT TYPES

* Edge-optimized API Endpoints
  + The default host name of an API Gateway API that isdeployed to the specified region while using a CloudFront distribution to facilitate client access typicall from across AWS regions. API requests are routed to the nearest CloudFront Point of Presence
* Regional API endpoints
  + The host name of an API that is deployed to the specified region and intended to serve clients such as EC2 instances in the same eAWS region. API requests are targeted directly to the region specific api gateway without going through any cloudfront distribution
    - You can apply latency based routing on regional endpoints to deploy an api to multiple regions using the same regional api endpoint configuration set the same custom domain name for each deployed api, and configure latency based DNS records in route 42to route client requests to the region that has the lowest latency.
* Private API endpoint
  + Allows a client to securely access private API resources inside a VPC. Private APIs are isolated from the public internet and they can only be accessed using VPC endpoints for API Gateway that have been granted access.

FEATURES

* API Gateway can execute Lambda code in your account, state tstp functions state machines or make calls to Elastic Beanstalk, EC2 or web services outside of aws with publicly accessible HTTP endpoints
* API Gateway helps you define plans that enter and restrict third party developer access to your APIs.
* API Gateway helps manage traffic to your backed systems by allowing you to set throttling rules based on the number of requests per second for each HTTP method in your APIs.
* You can set up a cache with customizable keys and time to live in seconds for your app data to avoid hitting you backend services for each request
* Run multiple versions of the same API simultaneously with API Lifecycle
* After you build, test, and deploy your APPIs, you can package them in an API Gateway usage plan and sell the plan as a software as a service (SaaS) product through AWS Marketplace
* API Gateway offers the ability to create, update and delete documentation associated with each portion of your api such as methods and resources.
* ALL of the APIs created expose HTTPS endpoints only. API Gateway does not support unencrypted (HTTP ) endpoints

MONITORING

* API Gateway console is integrated with CloudWatch, so you get backed performance metrics such as API calls, latency and errorates.
* You can set up custom alarms on API Gateway APIs.
* API Gateway can also log API execution errors to cloudwatch logs

SECURITY

* You can enable AWS WAF for your apis in amazon api gateway making it easier to protect your apis against common web exploits such as SQL injection and cross site scripting

**Question 17: Incorrect**

You developed a shell script which uses AWS CLI to create a new Lambda function. However, you received an InvalidParameterValueException after running the script.

What is the MOST likely cause of this issue?

* You have exceeded your maximum total code size per account.
* The AWS Lambda service encountered an internal error.
* You provided an IAM role in the CreateFunction API which AWS Lambda is unable to assume.(Correct)
* The resource already exists.(Incorrect)

#### 

To create a Lambda function, you need a deployment package and an execution role. The deployment package contains your function code. The execution role grants the function permission to use AWS services, such as Amazon CloudWatch Logs for log streaming and AWS X-Ray for request tracing. You can use the CreateFunction API via the AWS CLI or the AWS SDK of your choice.

A function has an unpublished version, and can have published versions and aliases. The unpublished version changes when you update your function's code and configuration. A published version is a snapshot of your function code and configuration that can't be changed. An alias is a named resource that maps to a version, and can be changed to map to a different version.

The InvalidParameterValueException will be returned if one of the parameters in the request is invalid. For example, if you provided an IAM role in the CreateFunction API which AWS Lambda is unable to assume. Hence, this option is the most likely cause of the issue in this scenario.

If you have exceeded your maximum total code size per account, the CodeStorageExceededException will be returned, which is why this option is incorrect.

If the resource already exists, the ResourceConflictException will be returned and not InvalidParameterValueException. Therefore, this option is also incorrect.

If the AWS Lambda service encountered an internal error, the ServiceException will be returned hence, this option is incorrect.

References:

<https://docs.aws.amazon.com/lambda/latest/dg/API_CreateFunction.html>

<https://docs.aws.amazon.com/cli/latest/reference/lambda/create-function.html>

Check out this AWS Lambda Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/>

**Question 18: Incorrect**

A company that sells smart security cameras uses an S3 bucket behind a CloudFront web distribution to store its static content, which they share to their customers around the world. They recently released a new firmware update that is intended to be used only by its premium customers. Any unauthorized access should be denied and the user authentication process must have a minimal latency.

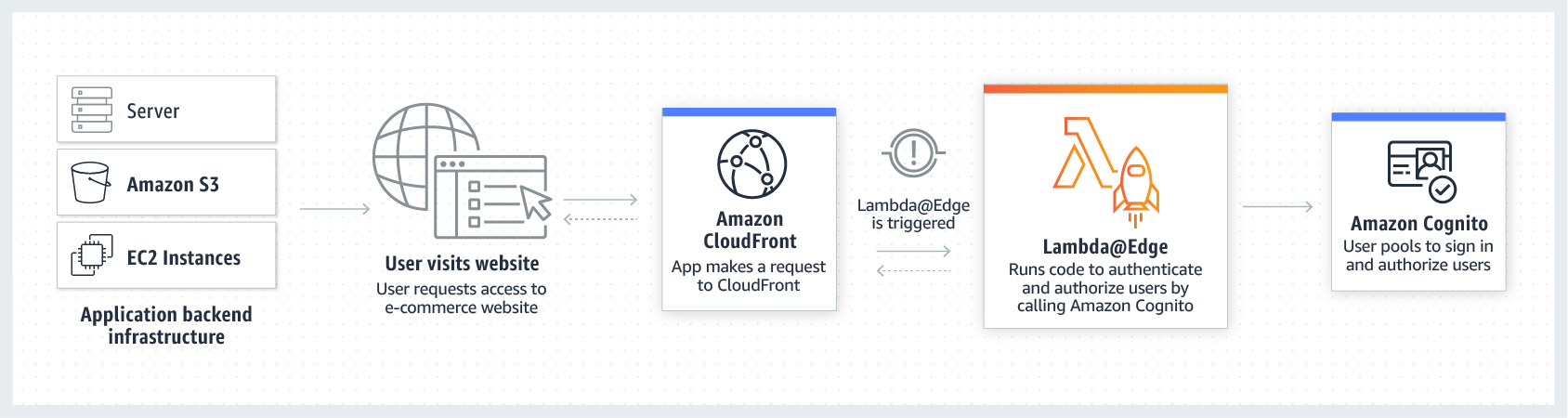
How can a developer refactor the current set up to achieve this requirement with the MOST efficient solution?

* Use the AWS Serverless Application Model (AWS SAM) and Amazon Cognito to authenticate the premium customers.​
* Use Lambda@Edge and Amazon Cognito to authenticate and authorize premium customers to download the firmware update.(Correct)
* ​Use Signed URLs and Signed Cookies in CloudFront to distribute the firmware update file.
* Restrict access to the S3 bucket only to premium customers by using an Origin Access Identity (OAI).(Incorrect)

#### 

Lambda@Edge is a feature of Amazon CloudFront that lets you run code closer to users of your application, which improves performance and reduces latency. With Lambda@Edge, you don't have to provision or manage infrastructure in multiple locations around the world. You pay only for the compute time you consume - there is no charge when your code is not running.

With Lambda@Edge, you can enrich your web applications by making them globally distributed and improving their performance — all with zero server administration. Lambda@Edge runs your code in response to events generated by the Amazon CloudFront content delivery network (CDN). Just upload your code to AWS Lambda, which takes care of everything required to run and scale your code with high availability at an AWS location closest to your end user.



You can use Lambda@Edge to help authenticate and authorize users for the premium pay-wall content on your website, filtering out unauthorized requests before they reach your origin infrastructure. For example, you can trigger a Lambda function to authorize each viewer request by calling authentication and user management service such as Amazon Cognito.

Hence, using Lambda@Edge and Amazon Cognito to authenticate and authorize premium customers to download the firmware update is the correct answer for this scenario.

Using the AWS Serverless Application Model (AWS SAM) and Amazon Cognito to authenticate the premium customers is incorrect because AWS SAM is just an open-source framework that you can use to build serverless applications on AWS. In this scenario, you have to integrate your CloudFront web distribution with Lambda@Edge and you can do this without using AWS SAM.

Restricting access to the S3 bucket only to premium customers by using an Origin Access Identity (OAI) is incorrect because OAI is primarily used to prevent your users from viewing your S3 files by simply using the direct S3 URL. This is used in conjunction with CloudFront signed URLs or signed cookies to limit access to files in your Amazon S3 bucket. Setting up an OAI is not enough to satisfy the requirement of the scenario.

Using Signed URLs and Signed Cookies in CloudFront to distribute the firmware update file is incorrect because although this solution provides a way to authenticate the premium users for the private content, the process of authentication has a significant latency in comparison to the Lambda@Edge solution. In this option, you have to refactor your application (which is deployed to a specific AWS region) to either create and distribute signed URLs to authenticated users or to send Set-Cookie headers that set signed cookies on the viewers for authenticated users. This will cause the latency, which could have been improved if the authentication logic resides on CloudFront edge locations using Lambda@Edge.

References:

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

<https://aws.amazon.com/lambda/edge/>

Check out these Amazon CloudFront and AWS Lambda Cheat Sheets:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/>

**Question 19: Incorrect**

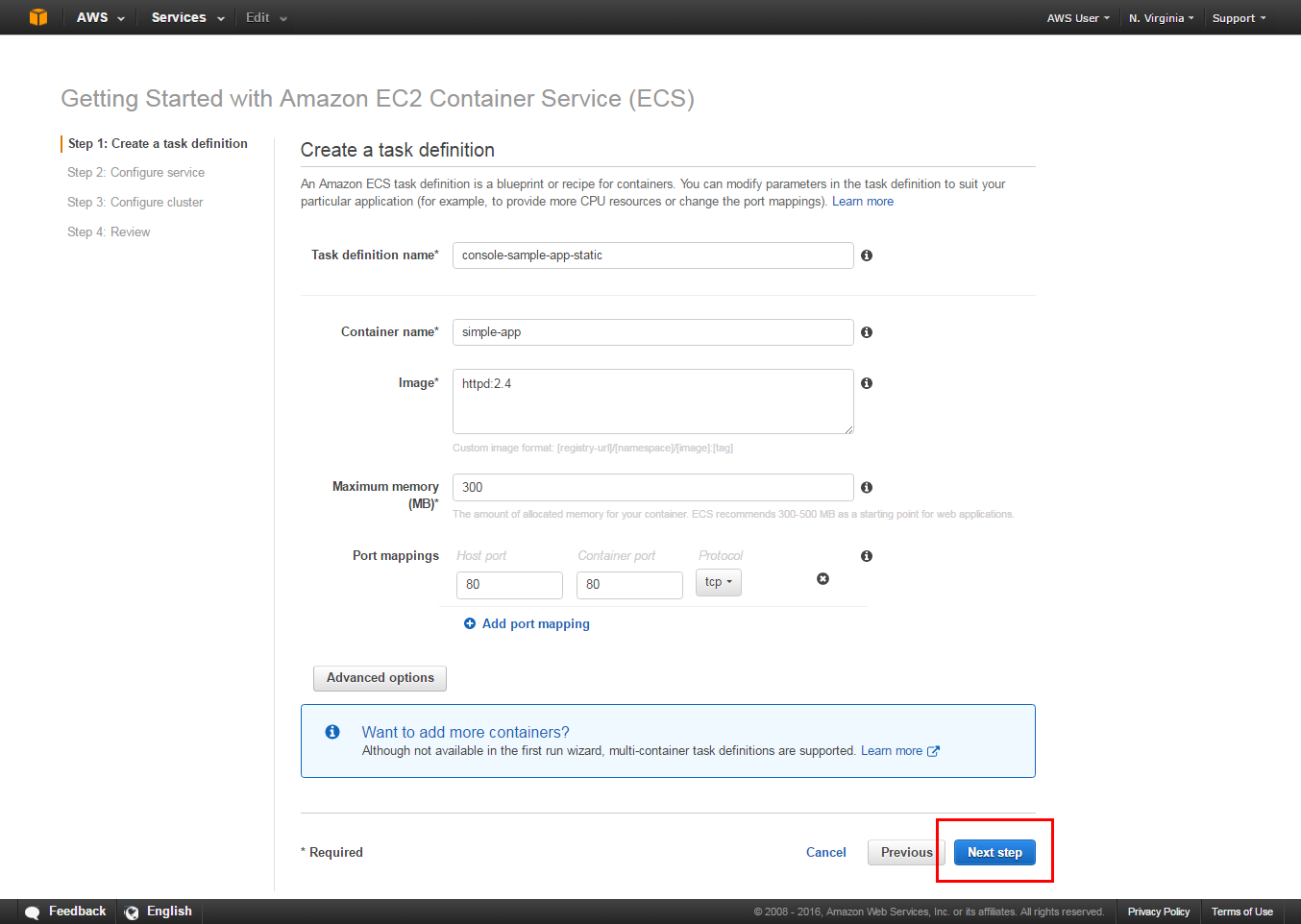
A developer is building a new Docker application using ECS. She needs to allow containers to access ports on the host container instance to send or receive traffic using port mapping.

Which component of ECS should the developer configure to properly implement this task?

* Service scheduler
* Container Agent(Incorrect)
* Task definition(Correct)
* ​Container instance

#### 

Port mappings allow containers to access ports on the host container instance to send or receive traffic. Port mappings are specified as part of the container definition which can be configured in the task definition.



For task definitions that use the awsvpc network mode, you should only specify the containerPort. The hostPort can be left blank or it must be the same value as the containerPort.

Port mappings on Windows use the NetNAT gateway address rather than localhost. There is no loopback for port mappings on Windows, so you cannot access a container's mapped port from the host itself.

Hence, the correct component that the developer should configure is Task Definition.

Service scheduler is incorrect because this only provides you the ability to run tasks manually (for batch jobs or single run tasks), with Amazon ECS placing tasks on your cluster for you. The service scheduler is ideally suited for long running stateless services and applications but not to configure port mappings.

Container instance is incorrect because this is just an Amazon EC2 instance that is running the Amazon ECS container agent and has been registered into a cluster. When you run tasks with Amazon ECS, your tasks using the EC2 launch type are placed on your active container instances. However, you can't manually configure the port mappings directly on your container instances but through task definitions.

Container Agent is incorrect because this only allows container instances to connect to your cluster. The Amazon ECS container agent is included in the Amazon ECS-optimized AMIs, but you can also install it on any Amazon EC2 instance that supports the Amazon ECS specification. Same as the other incorrect options, you can't configure port mappings with this component.

References:

<https://docs.aws.amazon.com/AmazonECS/latest/APIReference/API_PortMapping.html>

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html#container_definition_portmappings>

Check out this Amazon ECS Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/>

AMAZON ELASTIC CONTAINER SERVICE ( AMAZON ECS )

* A container management service to run, stop, and manage Docker containers on a cluster
* ECS can be used to create a consistent deployment and build experience, manage, and scale batch and **Extract Transform Load (ETL)** workloads, and build sophisticated application architectures on a microservice model.
* Amazon ECS is a regional service

FEATURES

* You can create ECS clusters within a new or existing VPC
* After a cluster is up and running you can define task definitions and services that specify which Docker Container images to run across you clusters
* AWS compute SLA guarantees a Monthly Uptime percentage of at least 99.99% for Amazon ECS

COMPONENTS

* Containers and images
  + Your application components must be architected to run in  **containers**
    - Containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc.
  + Containers are created from a read only template called an image
  + Images are typically built from a **Dockerfile, a** plain text file that specifies all of the components that are included in the container. THese images are then stored in a **registry** from which they can be downloaded and run on you cluster
  + When you launch a container instance, you have the option of passing user data to the instance. The data can be used to perform common automated configuration tasks and even run scripts when the instance boots
  + Docker volume can be a local instance store volume, EBS volume or EFS volume. Connect Docker containers to these volumes using Docker drivers and plugins
* Tasks components
  + **Task Definitions**
    - Specify various parameters for your application. It is a text file, in JSON format that describes one or more containers, up to a max of ten that form your application
    - Task definitions are split into separate parts
      * Task Family
        + The name of the task, and each family can have multiple revisions.
      * IAM task role
        + Specifies the permissions that containers in the task should have
      * Network mode
        + Determines how the networking is configured for your containers.
      * Container Definitions
        + Specify which image to use, how much CPU and memory the container is allocated and many more options
      * Volumes
        + Allows you to share data between containers and even persist the data on container instance when the containers are no longer running
      * Task Placement Constraints
        + Lets you customize how your tasks are placed within the infrastructure
      * Launch Types
        + Determines which infrastructure your tasks use.
* Tasks and Scheduling
  + **Task**
    - Is the instantiation of a task definition within a cluster. After you have created a task definition for your application , you can specify the number of tasks that will run on you cluster.
    - Each task that uses the Fargate launch type has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task
  + **Task Scheduler**
    - Is responsible for placing tasks within your cluster. There are several different scheduling options available
      * REPLICA
        + Places and maintains the desired number of tasks across your cluster. By default the service scheduler spreads tasks across AZs. YOu can use task placement strategies and constraints to customize task placement decisions
      * DAEMON
        + Deploys exactly one task on each active container instance that meets all of the task placement constraints that you specify in your cluster. When using this strat, there is no need to specify a desired number of tasks, a task placement strategy, or use service auto scaling policies
    - You can upload a new version of your application task definition and the ECS scheduler automatically starts new containers using the updated image and stops containers running the previous versions.

CLUSTERS

* When you run tasks using ECS, you place them in a **cluster**
  + A logical grouping of resources
  + Clusters are region specific
  + Contain tasks using both the Fargate And EC2 launch types
  + When using the EC2 launch type, then your clusters are a group of containers instances you manage. THese cluster can contain multiple different container instances types, but each container instance may only be a part of one cluster at a time
  + Before you can delete a cluster, you must delete the services and deregister the container instances inside that cluster.

SERVICES

* ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in a cluster
* In addition to maintaining the desired count of tasks in your service you can optionally run your service behind a load balancer
* There are two types of deployment strategies in ECS
  + **Rolling Update**
    - This involves the service scheduler replacing the current running versions of the container with the latest version.
    - The number of tasks ECS adds or removes from the service during a rolling update is controlled by the deployment configuration which consists of the minimum and max number of tasks allowed during a service deployment
  + **Blue / Green Deployment with AWS CodeDeploy**
    - This deployment type allows you to verify a new deployment of a service before sending production traffic to it.
    - The service must be configured to use either an application load balancer or network load balancer.

CONTAINER AGENT

* It runs on each infrastructure resource within an ECS cluster
* It sends information about the resource’s current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS
* Container agent is only supported on Amazon EC2 instances
* You can attach multiple target groups to your Amazon ECS services that are running on either Amazon EC2 or AWS fargate. This allows you to maintain a single ECS service that can serve traffic from both internal and external load balancers and support multiple path based routing rules and applications that need to expose more than one port.

AWS FARGATE

* YOu can use Fargate with ECS to run containers without having to manage servers or clusters of EC2 instances
* You no longer have to provision, configure, or scale cluster of virtual machines to run containers
* Fargate only supports container images hosted on Elastic Container Registry (ECR or Docker Hub)
* Usable in only select regions

TASK DEFINITIONS FOR FARGATE LAUNCH TYPES

* Require that the network mode is set to awsvpc. The awsvpc network mod provides each task with its own elastic network interface
* Require that you specify CPU and memory at the task level
* Only support the awslogs log driver for the log configuration. This configures your Fargate tasks to send log information to amazon cloudwatch logs
* Task storage is ephemeral. After a fargate task stops the storage is deleted
* Put multiple containers in the same task definition if:
  + Containers share a common lifecycle
  + Containers are required to be run on the same underlying host
  + You want your containers to share resources
  + You r containers share data volumes
* Otherwise define your containers in separate tasks definitions so that you can scale, provision and deprovision them separately

TASK DEFINITIONS FOR EC2 LAUNCH TYPES

* Create task definitions that group the containers that are used for a common purpose, and separate the different components into multiple task definitions
* After you have your task definitions, you can create services from them to maintain the availability of your desired tasks.
* For EC2 tasks, the following are the types of data volumes that can be used:
  + Docker volumes
  + Bind mounts
* Private repositories are only supported by the EC2 Launch types

TASK PLACEMENT STRATEGIES

* Binpack
  + Place tasks based on the least available amount of CPU or memory. This minimizes the number of instances in use
* Random
  + Place tasks at random
* Spread
  + Place tasks evenly based on the specified value. Accepted values are instance ID ( or host, which has the same effect), or any platform or custom attribute that is applied to a container instance, such as attribute: ecs.availability-zone. Service tasks are spread based on the tasks from that service. Standalone tasks are spread based on the tasks from the same task group

TASK PLACEMENT

* When a task task that uses the EC2 launch type is launched ,amazon ecs must determine where to place the task based on specific requirements specified in the task definition, such as CPU and memory. Similarly when you scale down the task count, amazon ecs must determine which tasks to terminate. You can apply task placement strategies and constraints to customize how AAmazon ECSA places and terminate tasks. Task placement strategy and constraints are not supported for tasks using the Fargeate launch type by default Fargate tasks are spread across AZs
* **A task placement strategy** is an algorithm for selecting instances for task placement or tasks for termination. For example amazon ecs can select instance at random or it can select instance such that tasks are distributed evenly across a group of instances.
* A task placement constraint is a rule that is considered during task placement. For example you can use constraints to place tasks based on AZ or instance type. You can also associate **attributes**, which are name/value pairs, with your container instances and then sue a constraint to place tasks based on attributes

**Question 21: Incorrect**

An application hosted in an Auto Scaling group of On-Demand EC2 instances is used to process data polled from an SQS queue and the generated output is stored in an S3 bucket. To improve security, you were tasked to ensure that all objects in the S3 bucket are encrypted at rest using server-side encryption with AWS KMS–Managed Keys (SSE-KMS).

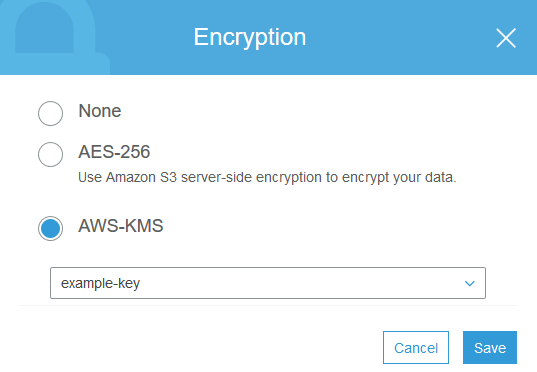
Which of the following is required to properly implement this requirement?

* Add a bucket policy which denies any s3:PostObject action unless the request includes the x-amz-server-side-encryption header.
* ​Add a bucket policy which denies any s3:PostObject action unless the request includes the x-amz-server-side-encryption-aws-kms-key-id header.
* Add a bucket policy which denies any s3:PutObject action unless the request includes the x-amz-server-side-encryption-aws-kms-key-id header.(Incorrect)
* ​Add a bucket policy which denies any s3:PutObject action unless the request includes the x-amz-server-side-encryption header.(Correct)

#### 

Server-side encryption is about protecting data at rest. AWS Key Management Service (AWS KMS) is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. AWS KMS uses customer master keys (CMKs) to encrypt your Amazon S3 objects. You use AWS KMS via the [AWS Management Console](https://console.aws.amazon.com/kms) or [AWS KMS APIs](https://docs.aws.amazon.com/kms/latest/APIReference/) to centrally create encryption keys, define the policies that control how keys can be used, and audit key usage to prove they are being used correctly. You can use these keys to protect your data in Amazon S3 buckets.

The first time you add an SSE-KMS–encrypted object to a bucket in a region, a default CMK is created for you automatically. This key is used for SSE-KMS encryption unless you select a CMK that you created separately using AWS Key Management Service. Creating your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.



Amazon S3 supports bucket policies that you can use if you require server-side encryption for all objects that are stored in your bucket. For example, you can set a bucket policy which denies permission to upload an object (s3:PutObject) to everyone if the request does not include the x-amz-server-side-encryption header requesting server-side encryption with SSE-KMS.

When you upload an object, you can specify the KMS key using the x-amz-server-side-encryption-aws-kms-key-id header which you can use to require a specific KMS key for object encryption. If the header is not present in the request, Amazon S3 assumes the default KMS key. Regardless, the KMS key ID that Amazon S3 uses for object encryption must match the KMS key ID in the policy, otherwise Amazon S3 denies the request.

Therefore, the correct answer in this scenario is to add a bucket policy which denies any s3:PutObject action unless the request includes the x-amz-server-side-encryption header.

Adding a bucket policy which denies any s3:PutObject action unless the request includes the x-amz-server-side-encryption-aws-kms-key-id header is incorrect because you have to use the x-amz-server-side-encryption header instead.

Adding a bucket policy which denies any s3:PostObject action unless the request includes the x-amz-server-side-encryption header is incorrect because you have to deny s3:PutObject and not the s3:PostObject action.

Adding a bucket policy which denies any s3:PostObject action unless the request includes the x-amz-server-side-encryption-aws-kms-key-id header is incorrect because you have to use the x-amz-server-side-encryption header instead. Moreover, you have to deny s3:PutObject and not the s3:PostObject action.

References:

<https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html>

<https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html>

Check out this Amazon S3 Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-s3/>

**Question 25: Incorrect**

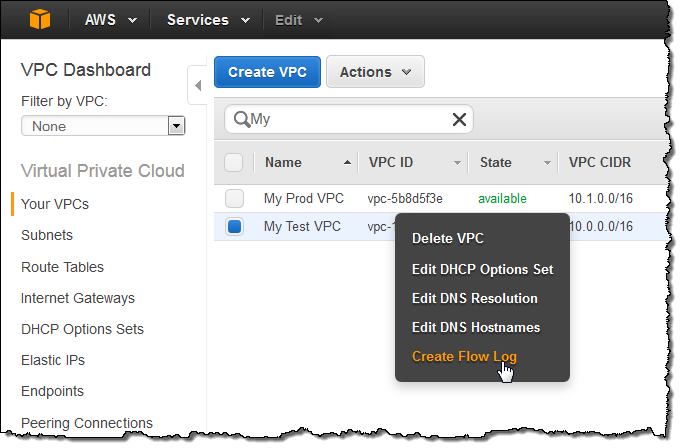
An online stock trading platform is hosted in an Auto Scaling group of EC2 instances with an Application Load Balancer in front to evenly distribute the incoming traffic. The developer must capture information about the IP traffic going to and from network interfaces in your VPC to comply with financial regulatory requirements.

Which of the following options should the developer do to meet the requirement?

* Use AWS Inspector to capture information about the IP traffic going to and from the network interfaces of your EC2 instances.
* Use CloudTrail logs to track all API calls and capture information about the IP traffic going to and from your VPC.(Incorrect)
* Create a flow log in your VPC.(Correct)
* Install and run the AWS X-Ray daemon to your EC2 instances using an instance metadata script.

#### 

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs and Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination.



Flow logs can help you with a number of tasks; for example, to troubleshoot why specific traffic is not reaching an instance, which in turn helps you diagnose overly restrictive security group rules. You can also use flow logs as a security tool to monitor the traffic that is reaching your instance. CloudWatch Logs charges apply when using flow logs, whether you send them to CloudWatch Logs or to Amazon S3.

Hence, you should create a flow log in your VPC to capture information about the IP traffic going to and from network interfaces in your VPC.

Using CloudTrail logs to track all API calls and capture information about the IP traffic going to and from your VPC is incorrect because although you can indeed use CloudTrail to track the API call, it can't capture information about the IP traffic of your VPC.

Installing and running the AWS X-Ray daemon to your EC2 instances using an instance metadata script is incorrect because you have to use a user data script and not a metadata. Alternatively, you can instrument your application which is running in an EC2 instance to capture the client's IP address. However, it is much easier to just enable VPC Flow Logs to meet the requirement.

Using AWS Inspector to capture information about the IP traffic going to and from the network interfaces of your EC2 instances is incorrect because this service is just an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. It doesn't have the ability to capture IP traffic of your VPC.

References:

<https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html>

<https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon-ec2.html>

<https://docs.aws.amazon.com/xray/latest/devguide/xray-concepts.html>

Check out these Amazon VPC and AWS X-Ray Cheat Sheets:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-vpc/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-x-ray/>

**Question 26: Incorrect**

An application, which already uses X-Ray, generates thousands of trace data every hour. The developer wants to use a filter expression which will limit the results based on custom attributes or keys that he specifies.

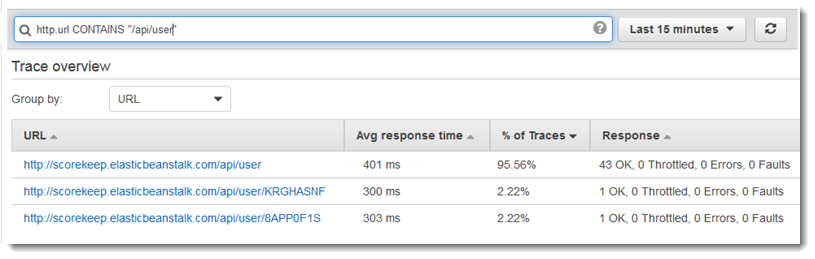
How should the developer refactor the application in order to filter the results in the X-Ray console?

​

* Create a new sampling rule based on the custom attributes.(Incorrect)
* Add the custom attributes as metadata in your segment document.
* Include the custom attributes as new segment fields in the segment document.
* Add the custom attributes as annotations in your segment document.(Correct)

#### 

Even with sampling, a complex application generates a lot of data. The AWS X-Ray console provides an easy-to-navigate view of the service graph. It shows health and performance information that helps you identify issues and opportunities for optimization in your application. For advanced tracing, you can drill down to traces for individual requests, or use filter expressions to find traces related to specific paths or users.



When you instrument your application, the X-Ray SDK records information about incoming and outgoing requests, the AWS resources used, and the application itself. You can add other information to the segment document as annotations and metadata.

Annotations are simple key-value pairs that are indexed for use with [filter expressions](https://docs.aws.amazon.com/xray/latest/devguide/xray-console-filters.html). Use annotations to record data that you want to use to group traces in the console, or when calling the [GetTraceSummaries](https://docs.aws.amazon.com/xray/latest/api/API_GetTraceSummaries.html) API. X-Ray indexes up to 50 annotations per trace.

Metadata are key-value pairs with values of any type, including objects and lists, but that are not indexed. Use metadata to record data you want to store in the trace but don't need to use for searching traces. You can view annotations and metadata in the segment or subsegment details in the X-Ray console.

Hence, adding the custom attributes as annotations in your segment document is the correct answer.

Including the custom attributes as new segment fields in the segment document is incorrect because a segment field can't be used as a filter expression. You have to add the custom attributes as annotations to the segment document that you'll send to X-Ray, just as mentioned above.

Creating a new sampling rule based on the custom attributes is incorrect because sampling is primarily used to ensure efficient tracing and to provide a representative sample of the requests that your application serves.

Adding the custom attributes as metadata in your segment document is incorrect because metadata is primarily used to record custom data that you want to store in the trace but not for searching traces.

References:

<https://docs.aws.amazon.com/xray/latest/devguide/xray-concepts.html#xray-concepts-annotations>

<https://docs.aws.amazon.com/xray/latest/devguide/xray-console-filters.html>

Check out this AWS X-Ray Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-x-ray/>

Question 29: Incorrect

A developer is building an e-commerce application which will be hosted in an ECS Cluster. To minimize the number of instances in use, she must select a strategy which will place tasks based on the least available amount of CPU or memory.

Which of the following task placement strategy should the developer implement?

* distinctInstance​
* spread(Incorrect)
* binpack(Correct)
* ​random

#### Explanation

A task placement strategy is an algorithm for selecting instances for task placement or tasks for termination. Task placement strategies can be specified when either running a task or creating a new service.

Amazon ECS supports the following task placement strategies:

binpack - Place tasks based on the least available amount of CPU or memory. This minimizes the number of instances in use.

random - Place tasks randomly.

spread - Place tasks evenly based on the specified value. Accepted values are attribute key-value pairs, instanceId, or host.

The scenario states that the developer must select a task placement strategy which will place tasks based on the least available amount of CPU or memory. By using bin pack strategy with CPU as the field parameter, ECS is able to place tasks onto an instance with the least available amount of CPU first, before moving on to the other instances. Hence, the correct answer is to use the binpack task placement strategy.

random is incorrect because this will place the tasks randomly, rather than placing the tasks to the instances based on the least available amount of CPU or memory.

spread is incorrect because this will place tasks evenly to the instances based on a specified value.

distinctInstance is incorrect because this is not a valid task placement strategy, but a task placement constraint. This is primarily used as a constraint to place each task on a different container instance. It can be specified when either running a task or creating a new service.

References:

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-placement.html>

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-placement-strategies.html>

<https://aws.amazon.com/blogs/compute/amazon-ecs-task-placement/>

Check out this Amazon ECS Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/>

Question 30: Incorrect

A web application is running in an ECS Cluster and updates data in DynamoDB several times a day. The clients retrieve data directly from the DynamoDB through APIs exposed by Amazon API Gateway. Although API caching is enabled, there are specific clients that want to retrieve the latest data from DynamoDB for every API request sent.

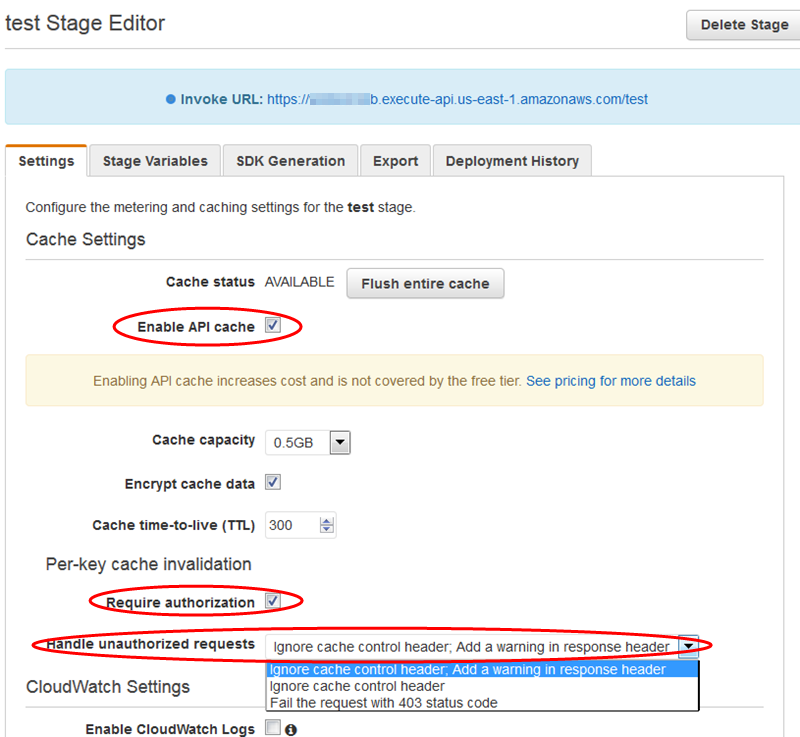
What should be done to only allow authorized clients to invalidate an API Gateway cache entry when submitting API requests? (Select TWO)

* The client must send a request which contains the Cache-Control: max-age=1 header.
* The client must send a request which contains the Cache-Control: max-age=0 header.(Correct)
* Provide your clients an authorization token from STS to query data directly from DynamoDB.​
* Tick the Require Authorization checkbox in the Cache Settings of your API via the console.
* (Correct)
* Modify the cache settings to retrieve the latest data from DynamoDB if the request header's authorization signature matches your API's trusted clients list.(Incorrect)

#### Explanation

A client of your API can invalidate an existing cache entry and reload it from the integration endpoint for individual requests.

The client must send a request that contains the **Cache-Control: max-age=0 header.** *The client receives the response directly from the integration endpoint instead of the cache*, provided that the client is authorized to do so. This replaces the existing cache entry with the new response, which is fetched from the integration endpoint.



Ticking the Require authorization checkbox ensures that not every client can invalidate the API cache. If most or all of the clients invalidate the API cache, this could significantly increase the latency of your API.

Hence, to only allow authorized clients to invalidate an API Gateway cache entry when submitting API requests, you can just tick the Require Authorization checkbox in the Cache Settings of your API via the console and instruct the client to send a request which contains the Cache-Control: max-age=0 header.

Instructing the client to send a request which contains the Cache-Control: max-age=1 header is incorrect because the value of the max-age should be 0 and not 1.

Providing your clients an authorization token from STS to query data directly from DynamoDB is incorrect because this will not enable your clients to invalidate the cache in API Gateway. Considering that your clients are using APIs to interact with DynamoDB, you should not provide them access to directly submit queries to your table but only through API Gateway.

Modifying the cache settings to retrieve the latest data from DynamoDB if the request header's authorization signature matches your API's trusted clients list is incorrect because this configuration can't be done. There is no feature in API Gateway Cache Settings which would allow you to make a list of authorized signatures that are allowed to invalidate cache entries.

References:

<https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html#invalidate-method-caching>

<https://aws.amazon.com/api-gateway/faqs/#Throttling_and_Caching>

Check out this Amazon API Gateway Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway/>

Question 35: Incorrect

Your application is processing one Kinesis data stream which has four shards, and each instance has one KCL worker. To scale up processing in your application, you reshard your stream to increase the number of open shards to six.

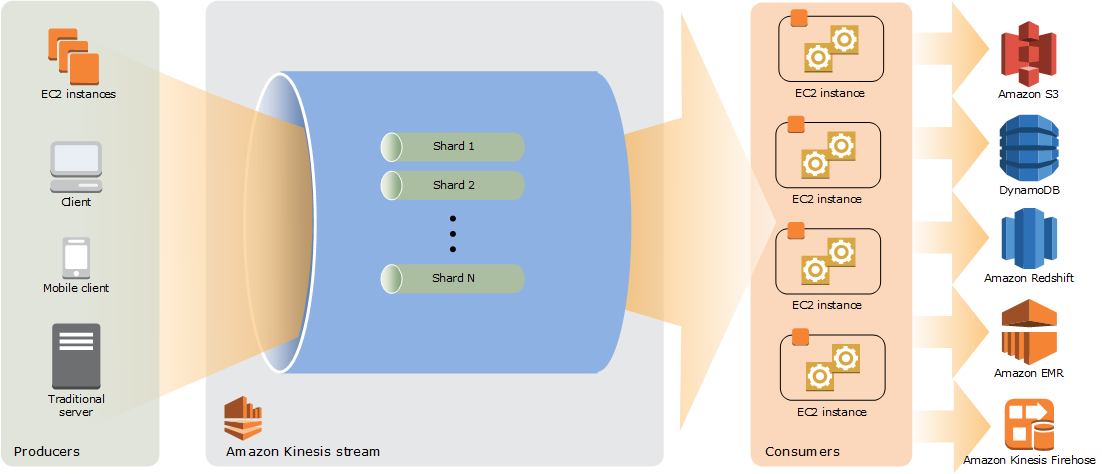
What is the MAXIMUM number of EC2 instances that you should launch to achieve optimum performance?

​

* 12(Incorrect)
* 3
* 6(Correct)
* 5

#### Explanation

Resharding enables you to increase or decrease the number of shards in a stream in order to adapt to changes in the rate of data flowing through the stream. The Kinesis Client Library (KCL) ensures that for every shard there is a record processor running and processing that shard. It also tracks the shards in the stream using an Amazon DynamoDB table.



Typically, when you use the KCL, you should ensure that the number of instances does not exceed the number of shards (except for failure standby purposes). Each shard is processed by exactly one KCL worker and has exactly one corresponding record processor, so you never need multiple instances to process one shard. However, one worker can process any number of shards, so it's fine if the number of shards exceeds the number of instances.

To scale up processing in your application, you should test a combination of these approaches:

- Increasing the instance size (because all record processors run in parallel within a process)

- Increasing the number of instances up to the maximum number of open shards (because shards can be processed independently)

- Increasing the number of shards (which increases the level of parallelism)

Thus, the maximum number of instances you can launch is 6, to match the number of open shards in a ratio of 1:1.

Although you can launch 3 instances in which each instance handles 2 shards, this is not the maximum number of instances you can deploy for your application. Hence, this option is incorrect. Take note that the maximum number of your instances is not half the number of open shards.

Just like the above option, you can also launch 5 instances in which each instance handles 3 shards. However, this is not the maximum number of instances you can launch. Keep in mind that the maximum number of your instances can be equal to the number of open shards of the Kinesis stream. Therefore, this option is also incorrect.

Launching 12 instances is incorrect because you should ensure that the number of instances does not exceed the number of open shards. The maximum number of instances that you should deploy is 6.

References:

<https://docs.aws.amazon.com/streams/latest/dev/key-concepts.html>

<https://docs.aws.amazon.com/streams/latest/dev/kinesis-record-processor-scaling.html>

Check out this Amazon Kinesis Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-kinesis/>

Kinesis Scaling, Resharding and Parallel Processing:

<https://tutorialsdojo.com/aws-cheat-sheet-kinesis-scaling-resharding-and-parallel-processing/>

AMAZON KINESIS

* Makes it easy to collec, process and analyze real time streaming data
* Kinesis can ingest real time data such as video audio application logs website clickstreams and IoT telemetry data for machine learning analytics and other applications

KINESIS VIDEO STREAMS

* A fully managed aws service that you can use to stream live video from device to the AWS cloud or build application for real time video processing or batch oriented video analytics
* BENEFITS
  + You can connect and stream from millions of devices
  + You can configure your kinesis video stream to durably store media data for custom retention periods kinesis video setams also generates index over the stored databased oin producer - generated or service side timestamps
  + Kinesis video streams is serverless, so there is no infrastructure to manage or setup
  + YOu can build real time and batch applications on data streams
  + Kinesis video streams enforces trasport layer security (TLS) based encryption on data streaming from devices and encrypts all data at rest using AWS KMS
* COMPONENTS
  + **Producer**
    - Any source that puts data into a kinesis video steam
  + Kinesis video stream
    - A resource that enables you to transport live video data, optionally store it, and make the data available for consumption both in real time and on a batch or ad hoc basis
    - Time-encoded data
      * Is any data in which the records are in a time series, and each record is related to its previous and next records
    - Fragment
      * Is a self contained sequence of frames The frames belonging to a fragment belonging to a fragment should have no dependency on any frames from other fragments
    - Ypon receiving the data from a producer, kinesis videos streams stores incoming media data as **chunks**
      * Each chunk consists of the actual media fragment, a copy of media metadata sent by producer, and kinesis video stream specific metadata such as fragment number, and server side and producer side timestamps
    - **Consumer**
      * Gets data such as fragments and frames, from a kinesis video stream to view the process or analyze it. Generally these consumers are called kinesis video streams applications.

KINESIS VIDEO STREAMS:

* APIs for you co creat enad manage streams and read or write data to and from a stream
* A console that support live and video-on-demand playback
* A set of producer libraries that you can use in your applicaiton code to etract data from your media sources and upload t oy our kinesis video stream

Video playbacks

* You can view a kinesis video stream using either
* HTTP Live streaming
  + You can use JLS for live playback
* GetMedia API
  + YOu use the Get media API to build your own applications to process Kinesis video streams. GetMedia is a real time API with low latency

METADATA

* Metadata is a mutable key-value pair. YOu can use it to describe the content of the fragment, embed associated sensor readings that need to be transferred along 3ith the actual fragment, or meet other custom needs.
* There are two modes in which the metadata can be embedded with fragments in a stream:
  + Nonpersistent:
    - you can affix metadata on an ad hoc basis to a fragments in a stream based on business specific criteria that have occured
  + Persistent
    - You can affix metadata to successive, consecutive fragments in a stream based on a continuing need

KINESIS DATA STREAM

* A massively scalable, highly durable data ingestion and processing service optimized for streaming data. You can configure hundreds of thousands of data producers to continuously put data into a kinesis data stream

CONCEPTS

* Data producers
  + An application that typically emits data records as they are generated to a kinesis data stream. Data producers assign partition keys to records Partition keys ultimately determine which shard ingests the data record for a data stream
* Data consumers
  + A distributed kinesis application or AWS service retrieving data from all shards in a stream as its is generated most data consumers are retrieving the most recent data in a shard, enabling real time analytics or handling of data
* Data stream
  + A logical grouping of shards. There are no bounds on the number of shards within a data stream. Adata stream will retain data for 24 hours, or up to 7 days when extended retention is enabled
* Shard
  + The base throughput unit of a kinesis data stream
    - A shard is an append only log and a unit of streaming capability.
      * A shard contains an ordered sequence of records by arrival time
    - Add or remove shards from your stream dynamically as your data throughput changes
    - One shard can ingest up to 1000 data records per seconds or 1MB/sec.
      * Add more shards to increase your ingestion capability
    - When consumers use enhanced fanout, one shard provides 1MB/sec dat inputa nd2MD/sec data output foreach data consuemer register to use enhanced fanout
    - When consumers do not use enhanced fan out a shard provides 1MB/sec of input and 2MB/ sec data output and this output is shared with any consumer not using enhanced fan-out
    - You specify the number of shards needed when you create a stream and can change the quantity at any time
* Data Record
  + A record is the unit of data stored ina kinesis stream.
  + A record is composed of a sequence number, partitionkey m and data blob
  + A data blob is the data of interest your data producer adds to a stream
  + The Max size of a data blob is 1MB
* Partition Key
  + A partition key is typically a meaningful identifier, such as a user ID or timestamp. It is specified by your data producers will put data into a kinesis data stream and useful for consumers as they can use the partition key to replay or build a history associated with the partition key.
  + The partition key is also used to segregate and route data records to different shards.
* Sequence number
  + A sequence number is a unique identifier for each data record.
  + Sequence number is assigned by kinesis data streams when a data producer calls putrecord or putrecords API to add data to a kinesis data stream
* Amazon Kinesis Agent
  + A Pre-built java application that offers an easy way to collect and send data to your amazon kinesis data stream
* Monitoring
  + You can monitor shard level metrics in kinesis data streams
  + You can monitor your data streams in amazon kinesis data streams using cloud watch, kinesis agent, kinesis libraries
  + Log api calls with cloudtrail≥.

LIMITS

* There is no upper limit on the number of shards you can have in a steam or account
* There is no upper limit on the number of streams you can have in an account
* A single shard can ingest up to 1MB pers second s( including partition keys) or 1,000 records per second forwrites
* The default shard limit is 500 shards for US east us wes, eu irelands
  + For all others its 200 shards.
* **Each shard can support up to five read transactions per second.**

KINESIS DATA FIREHOSE

* The easiest way to load streaming data into a data store and analytics tool
* It is a fully managed service that automatically scales to match the throughput of your data.
* If can also batch, compress and encrypt the data before loading it.

FEATURES

* It can capture, transform, and load streaming data into S3 Redshift Elasticsearch Service, and Splunk, enabling near real time analytics with existing business intelligence tools and dashboards being used today
* You can specify a batch size or atch interval to control how quickly data is uploaded to destinations. Additionally you can specify if data should be compressed
* Once launched your delivery streams automatically scale up and down to handle GB of data per second or more of input data rate, and maintain data latency at level you specify for the stream
* Kinesis Data firehose can convert the format of incoming data from JSON to parquet or ORC formats before storing the data in S3
* YOu can configure kinesis data firehose to prepare your streaming data before it is loaded to data stores. Kinesis data firehose provides pre-built lambda blueprints for converting common data sources such as apache logs and system logs to JSON and CSV formats. You can use these prebuilt blueprints without any change or customize them further or write your own custom functions.

CONCEPTS

* Kinesis Data Firehose Delivery Stream
  + The underlying entity of kinesis data firehose. You can use kinesis data firehose by creating a kinesis data fierehose delivery stream and then sending data to it
* Record
  + The data of interest that your dat producers sends to a kinesis data firehose delivery stream
  + A record can be as large as 1,000 KB
* Data Producers
  + Producers send records to kinesis data firehose delivery streams
* Buffer Size and Buffer Interval
  + Kinesis data firehose buffers incoming streaming data to a certain size of for a certain period of time before delivering it to destinations. Buffer size is in MBs and Buffer interval isi in seconds.

STREAM SOURCES

* You can send data to your kinesis data firehose delivery stream using different types of sources:
  + A kinesis data stream
  + The kinesis agent
  + Or the kinesis data hose API using the AWS SDK
* You can also use cloudwatch logs cloudwatch events or aws IOT oas your data source
* Some aws services can only send messages and events to a kinesis data firehose delivery stream that is in the same region

DATA Delivery and TRANSFORMATION

* Kinesis data firehose can invoke lambda functions to transform incoming source data and deliver the transformed data to destinations
* Kinesis data fire hose buffers incoming data up to 3mb by default
* If your lambda function invocation fails because of a network timeout or because you’ve reached the lambda function limit kinesis data firehose retried the invocation three times by default
* Kinesis data fire hose can convert the format of your input data from JSON to apache parquet or apache ORC before storing the data in S3 . Parquet and ORC are columnar data formats and save space and enable faster queries compared to row oriented formats like JSON
* Data Delivery Format:
  + Data Delivery to S3
    - Kinesis data firehose concatenates multiple incoming records based on buffering configuration of your delivery steam. It then delivers the records to S3 as ans S3 object
  + Redshift
    - First delivers incoming data to your S3 bucket in the format described earlier. Kinesis data firehose then issues an redshift COPY command to load the data from your S3 bucket to your redshift cluster
  + ElasticSearch
    - Buffers incoming records based on the buffering configuration of your delivery stream. It then generates an elasticsearch bulk request to index multiple records to your elasticsearch cluster
  + Splunk
    - Concatenates the bytes that you send
* Data delivery frequency
  + S3
    - Determined by the s3 **buffers size** and **buffer interval** value that you configured for your **delivery stream**
  + Redshift
    - The frequency of data COPY operations from S3 to redshift is determined by how fast your redshift cluster cann finish the COPY command.
  + ElasticSearch
    - Determined by the **Elasticsearch buffers size** and **buffer interval** values that you configured for your deliveries stream
  + Splunk
    - The buffer size is 4MB and the buffer interval is 60 seconds

LIMITS

* By default, each account can have up to 50 kinesis data firehose delivery stream per region
* The max size of a record sent to kinesis data firehose before base64 encoding is 1000kb

KINESIS DATA ANALYTICS (SQL or JAVA)

* Analyze streaming data, gain actionable insights ,and respond to your business and customer needs in real time. yOu can quickly build SQL queries and Java applications using built in templat4es and operators for common processing functions to organize, transform, aggregate and analyze data at any scale.

GENERAL FEATURES

* Supports standard ANSI SQL
* Integrates with Kinesis data streams and kinesis data firehose so that you can readily ingest streaming data
* SQL applications in Analytics supports two types of inputs
  + Streaming data sources
    - Continuously generated data that is read into your application for processing
  + Reference data source
    - Static data that your application uses to enrich data coming in from streaming sources.
* Provides an easy to use schema editor to discover and edit the structure of the input data
  + The wizard automatically recognizes standard data formats such as JSON and CSV
* Offers functions optimized for stream processing so that you can easily perform advanced analytics such as anomaly detection and top-K analysis on your streaming data

SPECIAL JAVA FEATURES

COMPONENTS

* Input is the streaming sources for your application, In the input configuration you map the streaming sources to an in-application data streams
* Application code
  + A series of SQL statements that process input and produce output
* You can create one or more in applications teams to store the output
  + You can optionally configure an application output to persist data from specific insapplication streams to an external destination
* In Application data stream
  + Is an entity that continuously stores data in your application for you to perform processing
* Kinesis Data analytics provisions capacity in the form of :
* Kinesis Processing Units(KPU)
  + Single KPU provides you with the memory (4GB) and corresponding computing and networking

LIMITS

* The SQL code in an application is limited to 100KB
* The services ( for both SQL and JAVA) are only available in specific AWS regions
* You can create up to 50 kinesis data analytics applications per AWS region in your account
* The number of kinesis processing units (KPU) are limited to eight.

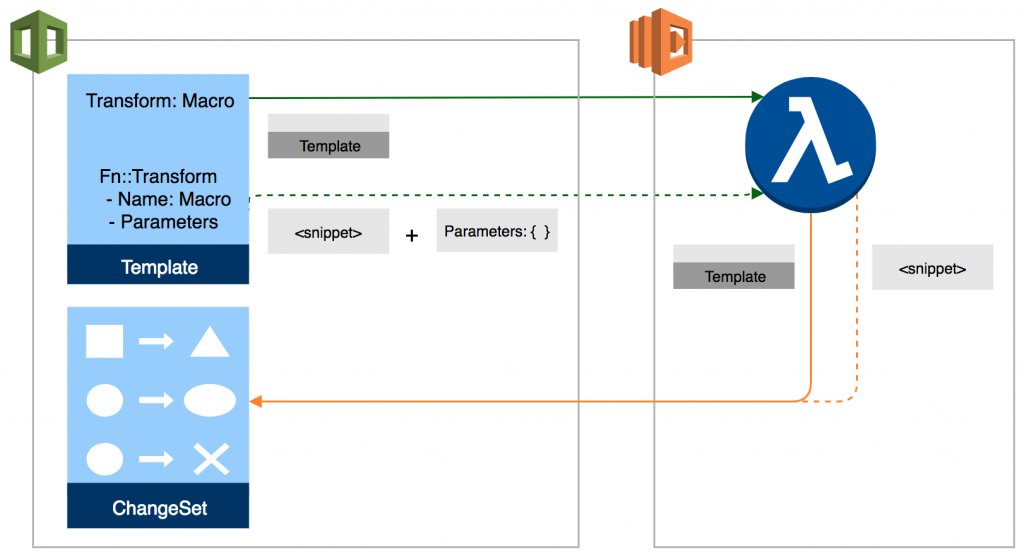
Question 40: Incorrect

You are deploying a serverless application composed of Lambda, API Gateway, CloudFront, and DynamoDB using CloudFormation. The AWS SAM syntax should be used to declare resources in your template which requires you to specify the version of the AWS Serverless Application Model (AWS SAM).

Which of the following sections is required, aside from the Resources section, that should be in your CloudFormation template?

* Parameters
* Transform(Correct)
* Mappings
* Format Version(Incorrect)

For serverless applications (also referred to as Lambda-based applications), the optional Transform section specifies the version of the AWS Serverless Application Model (AWS SAM) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it is processed.



This section specifies one or more macros that AWS CloudFormation uses to process your template. The Transform section builds on the simple, declarative language of AWS CloudFormation with a powerful macro system.

You can declare one or more macros within a template. AWS CloudFormation executes macros in the order that they are specified. When you create a change set, AWS CloudFormation generates a change set that includes the processed template content. You can then review the changes and execute the change set.

AWS CloudFormation also supports the AWS::Serverless and AWS::Include transforms, which are macros hosted by AWS CloudFormation. AWS CloudFormation treats these transforms the same as any macros you create in terms of execution order and scope.

Therefore, the Transform section should be the correct one to be added your template.

Mappings section is incorrect because this is just a literal mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table.

Parameters section is incorrect because this only contains the values that will be passed to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template but this is not used to specify the AWS SAM version.

Format Version section is incorrect because this just refers to the AWS CloudFormation template version that the template conforms to, and not the version of the AWS Serverless Application Model (AWS SAM)

References:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html>

<https://aws.amazon.com/blogs/aws/cloudformation-macros/>

Check out this AWS CloudFormation Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/>

Check out this AWS Serverless Application Model Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-serverless-application-model-sam/>

Question 46: Incorrect

A Lambda function has been integrated with DynamoDB Streams as its event source. There has been a new version of the function that needs to be deployed using CodeDeploy where the traffic must be shifted in two increments. It should shift 10 percent of the incoming traffic to the new version in the first increment and then the remaining 90 percent should be deployed five minutes later.

Which of the following deployment configurations is the MOST suitable to satisfy this requirement?

​

* Canary(Correct)
* All-at-once​
* Linear(Incorrect)
* Rolling with additional batch

#### Explanation

CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, serverless Lambda functions, or Amazon ECS services. CodeDeploy can deploy application content that runs on a server and is stored in Amazon S3 buckets, GitHub repositories, or Bitbucket repositories. CodeDeploy can also deploy a serverless Lambda function. You do not need to make changes to your existing code before you can use CodeDeploy.

When you deploy to an AWS Lambda compute platform, the deployment configuration specifies the way traffic is shifted to the new Lambda function versions in your application.



In a Canary deployment configuration, the traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment. Hence, this is the correct answer which will satisfy the requirement for the given scenario.

Linear is incorrect because this will cause the traffic to be shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.

All-at-once is incorrect because with this deployment configuration, the traffic is shifted from the original Lambda function to the updated Lambda function version all at once.

Rolling with additional batch is incorrect because this is only applicable in Elastic Beanstalk and not for Lambda.

References:

<https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-configurations.html>

<https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html>

Check out this AWS CodeDeploy Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-codedeploy/>

Question 48: Incorrect

A developer is working on a photo-sharing application which can automatically add filters to the high-resolution images uploaded by its users. For every new image that the user uploads, it would be sent to Amazon Kinesis and will be handled by an image processing application hosted in Lambda. The processed image would then be stored in an S3 bucket. If the upload was successful, the application will return a prompt telling the user that the upload is successful. However, the entire processing typically takes an average of 5 minutes to complete, which causes the application to become unresponsive.

Which of the following is the MOST suitable and cost-effective option which will prevent your application from being unresponsive?

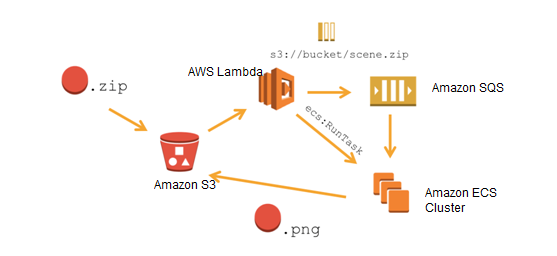
* Configure the application to asynchronously process the requests and change the invocation type of the Lambda function to Event.(Correct)
* Use a combination of Lambda and Step Functions to orchestrate service components and asynchronously process the requests.(Incorrect)
* Use AWS Serverless Application Model (AWS SAM) to allow asynchronous requests to your Lambda function.
* Configure the application to asynchronously process the requests and use the default invocation type of the Lambda function.

#### Explanation

AWS Lambda supports synchronous and asynchronous invocation of a Lambda function. You can control the invocation type only when you invoke a Lambda function (referred to as on-demand invocation). The following examples illustrate on-demand invocations:

* -Your custom application invokes a Lambda function.
* -You manually invoke a Lambda function (for example, using the AWS CLI) for testing purposes.

In both cases, you invoke your Lambda function using the [Invoke](https://docs.aws.amazon.com/lambda/latest/dg/API_Invoke.html) operation, and you can specify the invocation type as synchronous or asynchronous.



When you use AWS services as a trigger, the invocation type is predetermined for each service. You have no control over the invocation type that these event sources use when they invoke your Lambda function.

In the Invoke API, you have 3 options to choose from for the InvocationType:

RequestResponse (default) - Invoke the function synchronously. Keep the connection open until the function returns a response or times out. The API response includes the function response and additional data.

Event - Invoke the function asynchronously. Send events that fail multiple times to the function's dead-letter queue (if it's configured). The API response only includes a status code.

DryRun - Validate parameter values and verify that the user or role has permission to invoke the function.

Hence, the correct answer is to configure the application to asynchronously process the requests and change the invocation type of the Lambda function to Event.

Configuring the application to asynchronously process the requests and use the default invocation type of the Lambda function is incorrect because this will invoke your Lambda function synchronously. The default invocation type is RequestResponse which invokes the function synchronously and keeps the connection open until the function returns a response or times out.

Using AWS Serverless Application Model (AWS SAM) to allow asynchronous requests to your Lambda function is incorrect because AWS SAM just is an open-source framework that you can use to build serverless applications on AWS.

Using a combination of Lambda and Step Functions to orchestrate service components and asynchronously process the requests is incorrect because the AWS Step Functions service just lets you coordinate multiple AWS services into serverless workflows so you can build and update apps quickly. Although this can be a valid solution, it is not cost-effective since the application does not have a lot of components to orchestrate. Lambda functions can effectively meet the requirements in this scenario without using Step Functions by processing the requests asynchronously.

Reference:

<https://docs.aws.amazon.com/lambda/latest/dg/invocation-options.html>

<https://docs.aws.amazon.com/lambda/latest/dg/API_Invoke.html>

**Check out this AWS Lambda Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/>

Question 49: Incorrect

A DynamoDB table has several top-level attributes such as id, course\_id, course\_title, price, rating and many others. The database queries of your application returns all of the item attributes by default but you only want to fetch specific attributes such as the course\_id and price per request.

As the developer, how can you refactor your application to accomplish this requirement?

* Use projection expression(Correct)
* Use condition expressions
* Use filter expressions(Incorrect)
* Use expression attribute names

#### Explanation

To read data from a table, you use operations such as GetItem, Query, or Scan. DynamoDB returns all of the item attributes by default. To get just some, rather than all of the attributes, use a projection expression.

A projection expression is a string that identifies the attributes you want. To retrieve a single attribute, specify its name. For multiple attributes, the names must be comma-separated.

The following AWS CLI example shows how to use a projection expression with a GetItemoperation. This projection expression retrieves a top-level scalar attribute (Description), the first element in a list (RelatedItems[0]), and a list nested within a map (ProductReviews.FiveStar).

aws dynamodb get-item \

--table-name ProductCatalog \

--key '{"Id":{"N":"1"}}' \

--projection-expression "Description, RelatedItems[0], ProductReviews.FiveStar"

You can use any attribute name in a projection expression, provided that the first character is a-z or A-Z and the second character (if present) is a-z, A-Z, or 0-9. If an attribute name does not meet this requirement, you will need to define an expression attribute name as a placeholder.

Therefore, using projection expression is the correct answer in this scenario.

Using condition expressions is incorrect because this is primarily used to determine which items should be modified for data manipulation operations such as PutItem, UpdateItem, and DeleteItem calls.

Using expression attribute names is incorrect because this is a placeholder that you use in a projection expression, as an alternative to an actual attribute name. An expression attribute name must begin with a #, and be followed by one or more alphanumeric characters.

Using filter expressions is incorrect because it simply determines which items (and not the attributes) within the Query results should be returned to you. All of the other results are discarded. Take note that the scenario says that you have to fetch specific attributes and not specific items.

References:

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.ProjectionExpressions.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.html>

Question 50: Incorrect

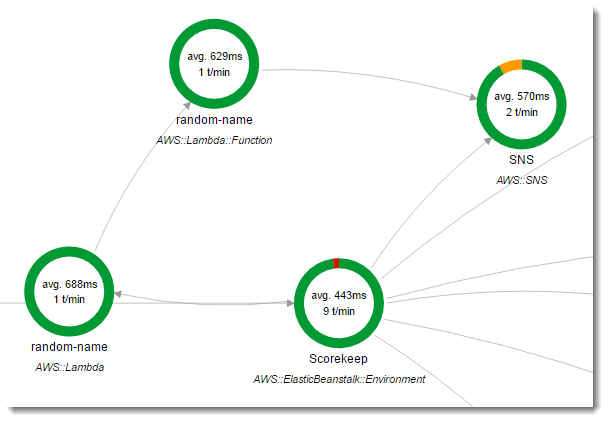
A recently deployed Lambda function has an intermittent issue in processing customer data. You enabled the active tracing option in order to detect, analyze, and optimize performance issues of your function using the X-Ray service.

Which of the following environment variables are used by AWS Lambda to facilitate communication with X-Ray? (Select TWO)

* AUTO\_INSTRUMENT
* AWS\_XRAY\_TRACING\_NAME(Incorrect​
* \_X\_AMZN\_TRACE\_ID(Correct)
* AWS\_XRAY\_DEBUG\_MODE(Incorrect)
* AWS\_XRAY\_CONTEXT\_MISSING(Correct)

#### Explanation

AWS X-Ray is an AWS service that allows you to detect, analyze, and optimize performance issues with your AWS Lambda applications. X-Ray collects metadata from the Lambda service and any upstream or downstream services that make up your application. X-Ray uses this metadata to generate a detailed service graph that illustrates performance bottlenecks, latency spikes, and other issues that impact the performance of your Lambda application.



AWS Lambda uses environment variables to facilitate communication with the X-Ray daemon and configure the X-Ray SDK.

\_X\_AMZN\_TRACE\_ID: Contains the tracing header, which includes the sampling decision, trace ID, and parent segment ID. If Lambda receives a tracing header when your function is invoked, that header will be used to populate the \_X\_AMZN\_TRACE\_ID environment variable. If a tracing header was not received, Lambda will generate one for you.

AWS\_XRAY\_CONTEXT\_MISSING: The X-Ray SDK uses this variable to determine its behavior in the event that your function tries to record X-Ray data, but a tracing header is not available. Lambda sets this value to LOG\_ERROR by default.

AWS\_XRAY\_DAEMON\_ADDRESS: This environment variable exposes the X-Ray daemon's address in the following format: IP\_ADDRESS:PORT. You can use the X-Ray daemon's address to send trace data to the X-Ray daemon directly, without using the X-Ray SDK.

Therefore, the correct answers for this scenario are the \_X\_AMZN\_TRACE\_ID and AWS\_XRAY\_CONTEXT\_MISSING environment variables.

AWS\_XRAY\_TRACING\_NAME is incorrect because this is primarily used in X-Ray SDK where you can set a service name that the SDK uses for segments.

AWS\_XRAY\_DEBUG\_MODE is incorrect because this is used to configure the SDK to output logs to the console without using a logging library.

AUTO\_INSTRUMENT is incorrect because this is primarily used in X-Ray SDK for Django Framework only. This allows the recording of subsegments for built-in database and template rendering operations.

References:

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-x-ray.html#viewing-lambda-xray-results>

<https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-nodejs-configuration.html>

<https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-python-configuration.html>

Check out this AWS X-Ray Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-x-ray/>

Instrumenting your Application with AWS X-Ray:

<https://tutorialsdojo.com/aws-cheat-sheet-instrumenting-your-application-with-aws-x-ray/>

Question 51: Incorrect

A serverless application composed of Lambda, API Gateway, and DynamoDB has been running without any issues for quite some time. As part of the IT compliance of the company, a developer was instructed to ensure that all of the new changes made to the items in DynamoDB are recorded and stored to another DynamoDB table in another region.

In this scenario, which of the following is the MOST ideal way to comply with the given requirements?

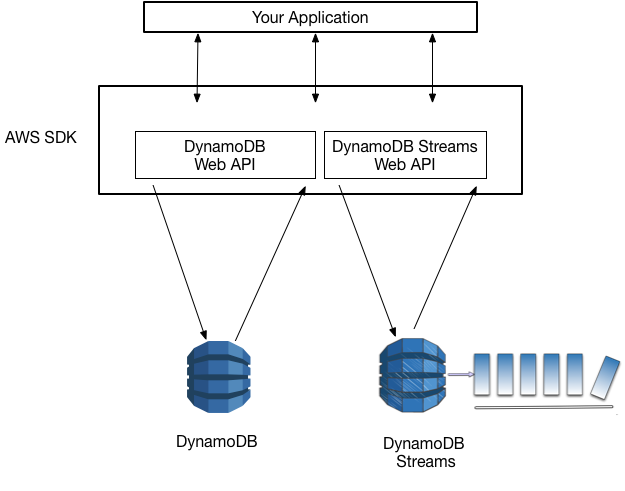
* Use DynamoDB Streams(Correct)
* Enable DynamoDB Point-in-Time Recovery
* Integrate CloudWatch Events to track the changes to DynamoDB(Incorrect)
* Set up DynamoDB Accelerator

#### Explanation

DynamoDB Streams enables solutions such as these, and many others. DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table, and stores this information in a log for up to 24 hours. Applications can access this log and view the data items as they appeared before and after they were modified, in near real time.

A DynamoDB stream is an ordered flow of information about changes to items in an Amazon DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attribute(s) of the items that were modified. A stream record contains information about a data modification to a single item in a DynamoDB table. You can configure the stream so that the stream records capture additional information, such as the "before" and "after" images of modified items.



Setting up DynamoDB Accelerator is incorrect because the DynamoDB Accelerator (DAX) feature simply takes the performance of the DynamoDB table to the next level with response times in microseconds for millions of requests per second for read-heavy workloads. You have to use DynamoDB Streams instead.

Enabling DynamoDB Point-in-Time Recovery is incorrect because this feature just helps protect your DynamoDB tables from accidental write or delete operations.

Integrating CloudWatch Events to track the changes to DynamoDB is incorrect because the CloudWatch Events service is not ideal to use for tracking changes to your DynamoDB table. The most appropriate feature that you should use is DynamoDB Streams.

References:

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.CrossRegionRepl.html>

Check out this Amazon DynamoDB Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/>

Check out this Amazon API Gateway Cheat Sheet:

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway/>

AWS Lambda Integration with Amazon DynamoDB Streams:

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda-integration-with-amazon-dynamodb-streams/>