**Multi Factor Authentication**

MFA adds extra security because it requires users to provide unique authentication from an AWS supported MFA mechanism in addition to their regular sign-in credentials when they access AWS websites or services:

* **Virtual MFA devices**. A software app that runs on a phone or other device and emulates a physical device. The device generates a six-digit numeric code based upon a time-synchronized one-time password algorithm. The user must type a valid code from the device on a second webpage during sign-in. Each virtual MFA device assigned to a user must be unique. A user cannot type a code from another user's virtual MFA device to authenticate. Because they can run on unsecured mobile devices, virtual MFA might not provide the same level of security as U2F devices or hardware MFA devices. We do recommend that you use a virtual MFA device while waiting for hardware purchase approval or while you wait for your hardware to arrive. For a list of a few supported apps that you can use as virtual MFA devices, see [Multi-Factor Authentication](http://aws.amazon.com/iam/details/mfa/). For instructions on setting up a virtual MFA device with AWS, see [Enabling a virtual multi-factor authentication (MFA) device (console)](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html).
* **U2F security key**. A device that you plug into a USB port on your computer. U2F is an open authentication standard hosted by the [FIDO Alliance](https://fidoalliance.org/). When you enable a U2F security key, you sign in by entering your credentials and then tapping the device instead of manually entering a code. For information on supported AWS U2F security keys, see [Multi-Factor Authentication](http://aws.amazon.com/iam/details/mfa/). For instructions on setting up a U2F security key with AWS, see [Enabling a U2F security key (console)](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_u2f.html).
* **Hardware MFA device**. A hardware device that generates a six-digit numeric code based upon a time-synchronized one-time password algorithm. The user must type a valid code from the device on a second webpage during sign-in. Each MFA device assigned to a user must be unique. A user cannot type a code from another user's device to be authenticated. For information on supported hardware MFA devices, see [Multi-Factor Authentication](http://aws.amazon.com/iam/details/mfa/). For instructions on setting up a hardware MFA device with AWS, see [Enabling a hardware MFA device (console)](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_physical.html).
* **SMS text message-based MFA**. A type of MFA in which the IAM user settings include the phone number of the user's SMS-compatible mobile device. When the user signs in, AWS sends a six-digit numeric code by SMS text message to the user's mobile device. The user is required to type that code on a second webpage during sign-in. Note that SMS-based MFA is available only for IAM users. You cannot use this type of MFA with the AWS account root user. For more information about enabling SMS text messaging-based MFA, see [PREVIEW – Enabling SMS text message MFA devices](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_sms.html).

# **Authenticate users using an Application Load Balancer**

[**PDF**](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/elb-ag.pdf#listener-authenticate-users)

[**Kindle**](https://www.amazon.com/dp/B0764757RN)

[**RSS**](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/application-load-balancer-release-notes.rss)

You can configure an Application Load Balancer to securely authenticate users as they access your applications. This enables you to offload the work of authenticating users to your load balancer so that your applications can focus on their business logic.

The following use cases are supported:

* Authenticate users through an identity provider (IdP) that is OpenID Connect (OIDC) compliant.
* Authenticate users through well-known social IdPs, such as Amazon, Facebook, or Google, through the user pools supported by Amazon Cognito.
* Authenticate users through corporate identities, using SAML, LDAP, or Microsoft AD, through the user pools supported by Amazon Cognito.

## Prepare to use an OIDC-compliant IdP

Do the following if you are using an OIDC-compliant IdP with your Application Load Balancer:

* Create a new OIDC app in your IdP. You must configure a client ID and a client secret.
* Get the following endpoints published by the IdP: authorization, token, and user info. You can locate this information in the well-known config.
* Allow one of the following redirect URLs in your IdP app, whichever your users will use, where DNS is the domain name of your load balancer and CNAME is the DNS alias for your application:
  + https://*DNS*/oauth2/idpresponse
  + https://*CNAME*/oauth2/idpresponse

## Prepare to use Amazon Cognito

Do the following if you are using Amazon Cognito user pools with your Application Load Balancer:

* Create a user pool. For more information, see [Amazon Cognito user pools](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html) in the *Amazon Cognito Developer Guide*.
* Create a user pool client. You must configure the client to generate a client secret, use code grant flow, and support the same OAuth scopes that the load balancer uses. For more information, see [Configuring a user pool app client](https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-client-apps.html) in the *Amazon Cognito Developer Guide*.
* Create a user pool domain. For more information, see [Adding a Domain name for your user pool](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-domain.html) in the *Amazon Cognito Developer Guide*.
* Verify that the requested scope returns an ID token. For example, the default scope, openid returns an ID token but the aws.cognito.signin.user.admin scope does not.
* To federate with a social or corporate IdP, enable the IdP in the federation section. For more information, see [Add social sign-in to a user pool](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-configuring-federation-with-social-idp.html) or [Add sign-in with a SAML IdP to a user pool](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-configuring-federation-with-saml-2-0-idp.html) in the *Amazon Cognito Developer Guide*.
* Allow the following redirect URLs in the callback URL field for Amazon Cognito, where DNS is the domain name of your load balancer, and CNAME is the DNS alias for your application (if you are using one):
  + https://*DNS*/oauth2/idpresponse
  + https://*CNAME*/oauth2/idpresponse
* Allow your user pool domain on your IdP app's callback URL. Use the format for your IdP. For example:
  + https://*domain-prefix*.auth.*region*.amazoncognito.com/saml2/idpresponse
  + https://*user-pool-domain*/oauth2/idpresponse

To enable an IAM user to configure a load balancer to use Amazon Cognito to authenticate users, you must grant the user permission to call the cognito-idp:DescribeUserPoolClient action.

## Prepare to use Amazon CloudFront

Enable the following settings if you are using a CloudFront distribution in front of your Application Load Balancer:

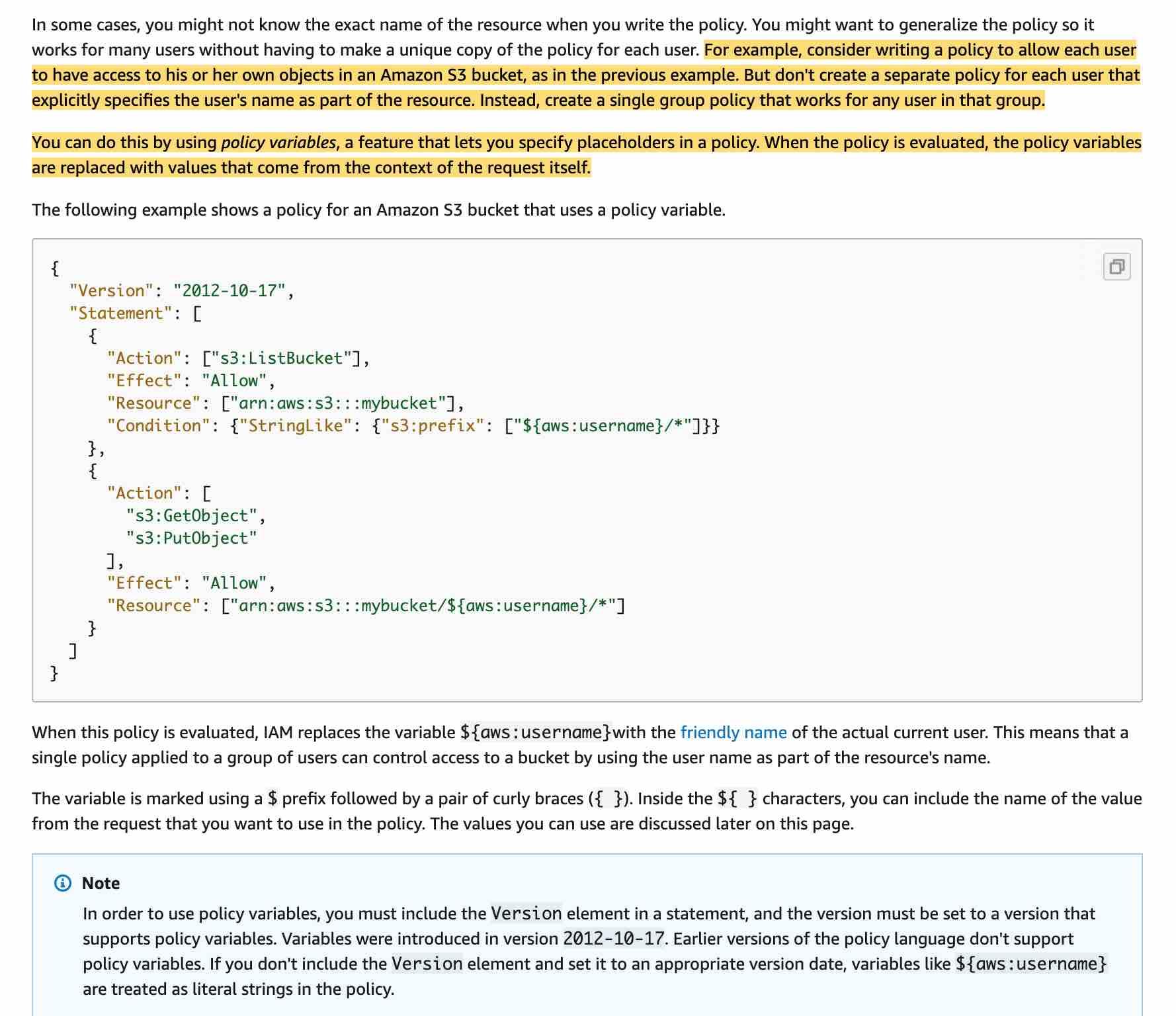
* Forward request headers (all) — Ensures that CloudFront does not cache responses for authenticated requests. This prevents them from being served from the cache after the authentication session expires. Alternatively, to reduce this risk while caching is enabled, owners of a CloudFront distribution can set the time-to-live (TTL) value to expire before the authentication cookie expires.
* Query string forwarding and caching (all) — Ensures that the load balancer has access to the query string parameters required to authenticate the user with the IdP.
* Cookie forwarding (all) — Ensures that CloudFront forwards all authentication cookies to the load balancer.

**IAM POLICY and parameters in it**

**IAM policy variables**

Instead of creating individual policies for each user, you can use policy variables and create a single policy that applies to multiple users (a group policy). Policy variables act as placeholders. When you make a request to AWS, the placeholder is replaced by a value from the request when the policy is evaluated.

As an example, the following policy gives each of the users in the group full programmatic access to a user-specific object (their own "home directory") in Amazon S3.

 via - <https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html>

Incorrect options:

**IAM policy principal** - You can use the Principal element in a policy to specify the principal that is allowed or denied access to a resource (In IAM, a principal is a person or application that can make a request for an action or operation on an AWS resource. The principal is authenticated as the AWS account root user or an IAM entity to make requests to AWS). You cannot use the Principal element in an IAM identity-based policy. You can use it in the trust policies for IAM roles and in resource-based policies.

**IAM policy condition** - The Condition element (or Condition block) lets you specify conditions for when a policy is in effect, like so - "Condition" : { "StringEquals" : { "aws:username" : "johndoe" }}. This can not be used to address the requirements of the given use-case.

**IAM policy resource** - The Resource element specifies the object or objects that the statement covers. You specify a resource using an ARN. This can not be used to address the requirements of the given use-case.

**AWS::Serverless::UserPool**

The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML.

SAM supports the following resource types:

AWS::Serverless::Api

AWS::Serverless::Application

AWS::Serverless::Function

AWS::Serverless::HttpApi

AWS::Serverless::LayerVersion

AWS::Serverless::SimpleTable

AWS::Serverless::StateMachine

UserPool applies to the Cognito service which is used for authentication for mobile app and web. There is no resource named UserPool in the Serverless Application Model.

Incorrect options:

**AWS::Serverless::Function** - This resource creates a Lambda function, IAM execution role, and event source mappings that trigger the function.

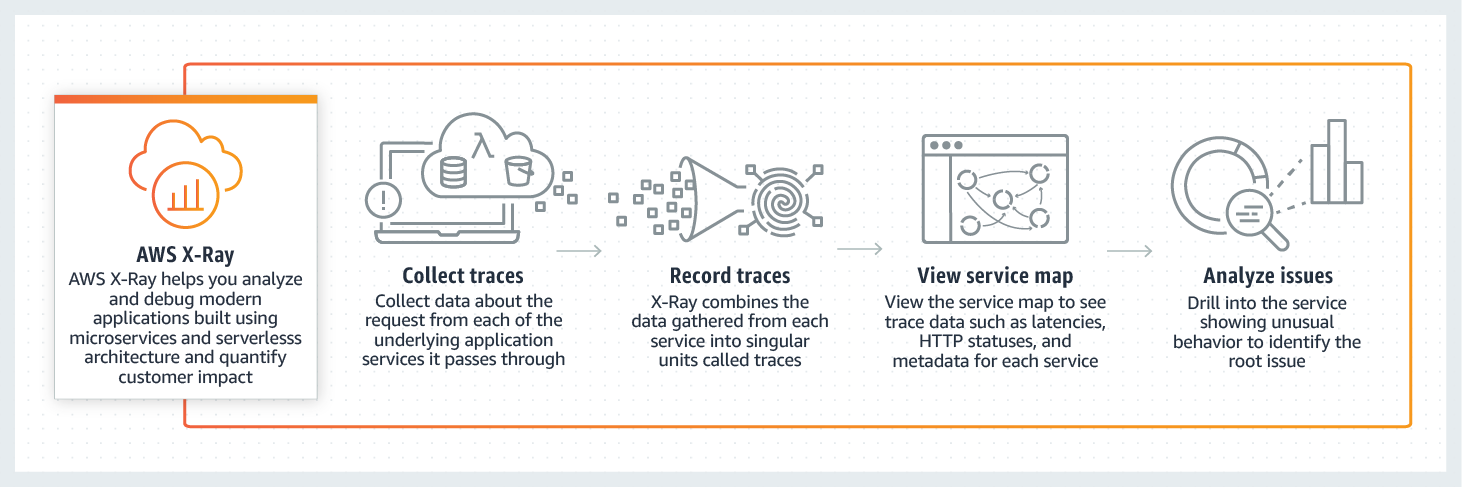
**AWS::Serverless::Api** - This creates a collection of Amazon API Gateway resources and methods that can be invoked through HTTPS endpoints. It is useful for advanced use cases where you want full control and flexibility when you configure your APIs.

**AWS::Serverless::SimpleTable** - This creates a DynamoDB table with a single attribute primary key. It is useful when data only needs to be accessed via a primary key.

**X-Ray**

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 collect data across AWS Accounts. The X-Ray agent can assume a role to publish data into an account different from the one in which it is running. This enables you to publish data from various components of your application into a central account.

How X-Ray Works:  via - <https://aws.amazon.com/xray/>

Incorrect options:

**VPC Flow Logs**: 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 is used to analyze network traces and helps with network security. Flow log data can be published to Amazon CloudWatch Logs or Amazon S3. You cannot use VPC Flow Logs to debug and trace data across accounts.

**CloudWatch Events**: Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in Amazon Web Services (AWS) resources. These help to trigger notifications based on changes happening in AWS services. You cannot use CloudWatch Events to debug and trace data across accounts.

**CloudTrail**: With CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. You can use AWS CloudTrail to answer questions such as - “Who made an API call to modify this resource?”. CloudTrail provides event history of your AWS account activity thereby enabling governance, compliance, operational auditing, and risk auditing of your AWS account. You cannot use CloudTrail to debug and trace data across accounts.

JWT token with web app

"Cognito User Pools"

After successful authentication, Amazon Cognito returns user pool tokens to your app. You can use the tokens to grant your users access to your own server-side resources, or to the Amazon API Gateway.

Amazon Cognito user pools implement ID, access, and refresh tokens as defined by the OpenID Connect (OIDC) open standard.

The ID token is a JSON Web Token (JWT) that contains claims about the identity of the authenticated user such as name, email, and phone\_number. You can use this identity information inside your application. The ID token can also be used to authenticate users against your resource servers or server applications.

"

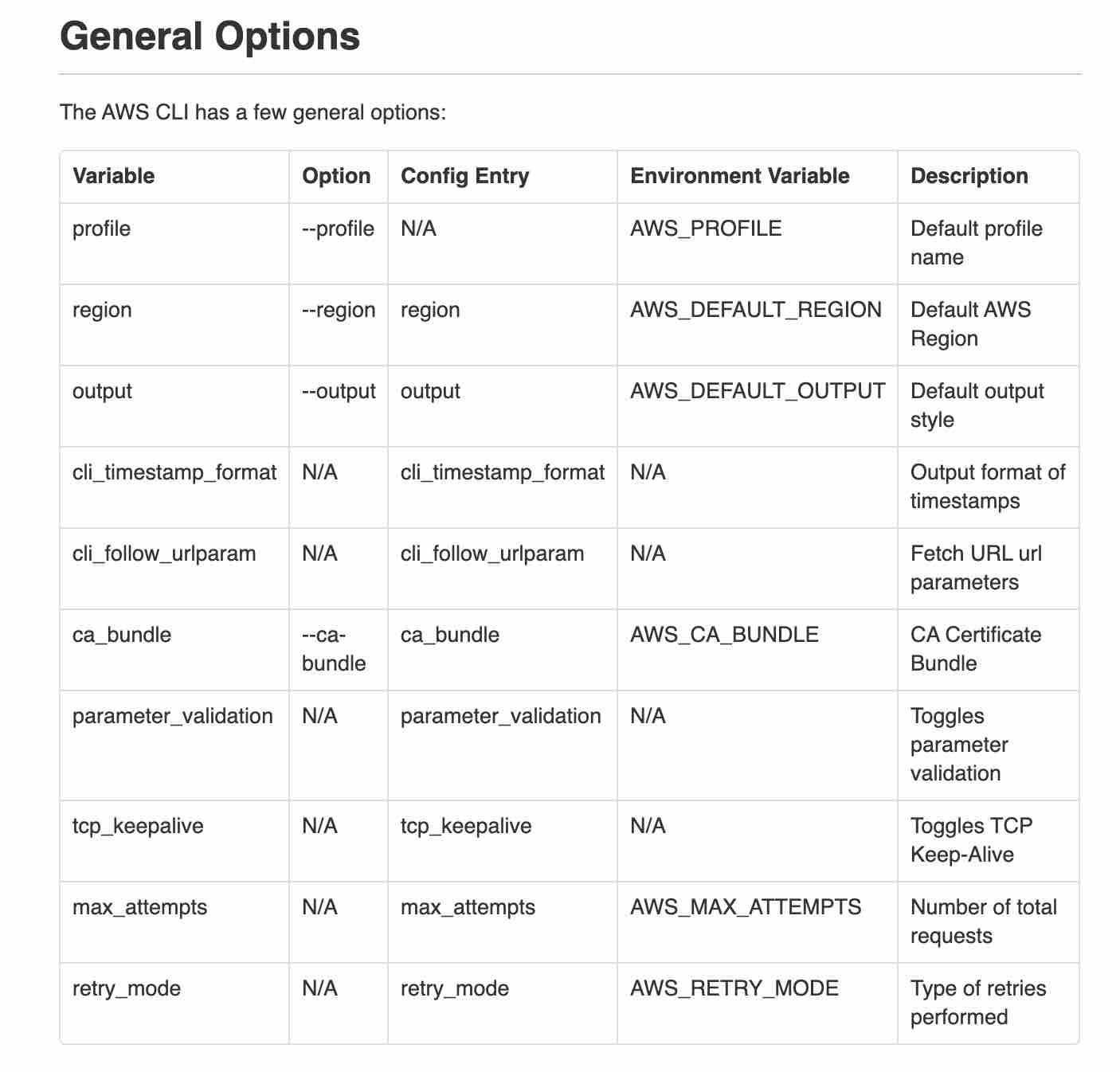
API Gateway" - If you are processing tokens server-side and using other programming languages not supported in AWS it may be a good choice. Other than that, go with a service already providing the functionality.

"Cognito Identity Pools" - You can use Identity pools to grant your users access to other AWS services. With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as the specific identity providers that you can use to authenticate users for identity pools.

"Cognito Sync" - Amazon Cognito Sync is an AWS service and client library that enables cross-device syncing of application-related user data. You can use it to synchronize user profile data across mobile devices and the web without requiring your own backend.

**How to execute commands from ec2 region1 to ec2 to region2 when access is there**

**Use the --region parameter**: If the region parameter is not set, then the CLI command is executed against the default AWS region.

You can also review all general options for AWS CLI:  via - <https://docs.aws.amazon.com/cli/latest/topic/config-vars.html#general-options>

Incorrect options:

**You need to override the default region by using aws configure** - This is not the most optimal way as you will have to change it again to reset the default region.

**You should create a new IAM user just for that other region** - This is not the most optimal way as you would need to manage two IAM user profiles.

**Use boto3 dependency injection** - With the CLI you do not use boto3. This option is a distractor.

API REQUEST response transform

**Use API Gateway Mapping Templates** - In API Gateway, an API's method request can take a payload in a different format from the corresponding integration request payload, as required in the backend. Similarly, vice versa is also possible. API Gateway lets you use mapping templates to map the payload from a method request to the corresponding integration request and from an integration response to the corresponding method response.

Suppose we have an API for managing fruit and vegetable inventory in the produce department of a supermarket. When a manager queries the backend for the current inventory, the server sends back the following response payload:

{

"department": "produce",

"categories": [

"fruit",

"vegetables"

],

"bins": [

{

"category": "fruit",

"type": "apples",

"price": 1.99,

"unit": "pound",

"quantity": 232

},

{

"category": "fruit",

"type": "bananas",

"price": 0.19,

"unit": "each",

"quantity": 112

},

{

"category": "vegetables",

"type": "carrots",

"price": 1.29,

"unit": "bag",

"quantity": 57

}

]

}

When the backend returns the query results shown above, the manager of the produce department might be interested in reading them, as follows:

{

"choices": [

{

"kind": "apples",

"suggestedPrice": "1.99 per pound",

"available": 232

},

{

"kind": "bananas",

"suggestedPrice": "0.19 per each",

"available": 112

},

{

"kind": "carrots",

"suggestedPrice": "1.29 per bag",

"available": 57

}

]

}

To enable this, we need to provide API Gateway with a mapping template to translate the data from the backend format like so:

#set($inputRoot = $input.path('$'))

{

"choices": [

#foreach($elem in $inputRoot.bins)

{

"kind": "$elem.type",

"suggestedPrice": "$elem.price per $elem.unit",

"available": $elem.quantity

}#if($foreach.hasNext),#end

#end

]

}

Incorrect options:

**Deploy an interceptor shell script** - This option has been added as a distractor.

**Use an API Gateway stage variable** - Stage variables are name-value pairs that you can define as configuration attributes associated with a deployment stage of a REST API. They act like environment variables and can be used in your API setup and mapping templates. This feature is not useful for the current use case.

**Use a Lambda custom interceptor** - This is a made-up option. Lambda cannot intercept the response for the given use-case.

Limit permissions but cannot grant Permissions

**AWS Organizations Service Control Policy (SCP)** – Use an AWS Organizations Service Control Policy (SCP) to define the maximum permissions for account members of an organization or organizational unit (OU). SCPs limit permissions that identity-based policies or resource-based policies grant to entities (users or roles) within the account, but do not grant permissions.

**Permissions boundary** - Permissions boundary is a managed policy that is used for an IAM entity (user or role). The policy defines the maximum permissions that the identity-based policies can grant to an entity, but does not grant permissions.

**Access control list (ACL)** - Use ACLs to control which principals in other accounts can access the resource to which the ACL is attached. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document structure. ACLs are cross-account permissions policies that grant permissions to the specified principal.

**Resource-based policy** - Resource-based policies grant permissions to the principal that is specified in the policy. Principals can be in the same account as the resource or in other accounts. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies.

**Identity-based policy** - Help attach managed and inline policies to IAM identities (users, groups to which users belong, or roles). Identity-based policies grant permissions to an identity.

CFT Conditions

**Parameters**

Parameters enable you to input custom values to your CloudFormation template each time you create or update a stack. Please see this note to understand how to define a parameter in a template:

**Resources** - Resources section describes the resources that you want to provision in your AWS CloudFormation stacks. You can associate conditions with the resources that you want to conditionally create.

**Conditions** - You actually define conditions in this section of the CloudFormation template

**Outputs** - The optional Outputs section declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console. For example, you can output the S3 bucket name for a stack to make the bucket easier to find. You can associate conditions with the outputs that you want to conditionally create.

**Common database but unique elastic cache**

**ElastiCache defined in .ebextensions/** - Any resources created as part of your .ebextensions is part of your Elastic Beanstalk template and will get deleted if the environment is terminated.

**ElastiCache defined in .ebextensions/** - Any resources created as part of your .ebextensions is part of your Elastic Beanstalk template and will get deleted if the environment is terminated.

**ElastiCache bundled with the application source code** - ElastiCache is an AWS service and cannot be bundled with the source code.

**RDS database defined in .ebextensions/** - The lifetime of the RDS instance gets tied to the lifetime of the Elastic Beanstalk environment, so this option is incorrect.

**ElastiCache database defined externally and referenced through environment variables** - For the given use-case, the client is fine with losing user session data and hence defining it in .ebextensions/ is more appropriate.

AMAZON Kinesis Data Streams vs DATAFirhose

**AWS Kinesis Data Streams**

Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data collected is available in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more

**AWS Kinesis Data Firehose** - Amazon Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you’re already using today. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security. As Kinesis Data Firehose is used to load streaming data into data stores, therefore this option is incorrect.

Please remember that Kinesis Data Firehose is used to load streaming data into data stores (Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk) whereas Kinesis Data Streams provides support for real-time processing of streaming data. It provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple downstream Amazon Kinesis Applications.

**Route 53 Routing Policy:**

**When you create a record, you choose a routing policy, which determines how Amazon Route 53 responds to queries:**

* **Simple routing policy – Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website.**
* **Failover routing policy – Use when you want to configure active-passive failover.**
* **Geolocation routing policy – Use when you want to route traffic based on the location of your users.**
* **Geoproximity routing policy – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another.**
* **Latency routing policy – Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency.**
* **Multivalue answer routing policy – Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.**
* **Weighted routing policy – Use to route traffic to multiple resources in proportions that you specify.**

**CFT use one sstack from another**

**Accesing network stack from webapp(it keeps two different logic seperately)**

# Walkthrough: Refer to resource outputs in another AWS CloudFormation stack

[**PDF**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-ug.pdf#walkthrough-crossstackref)

[**Kindle**](https://www.amazon.com/dp/B07641FR9Q)

[**RSS**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-cloudformation-release-notes.rss)

**To export resources from one AWS CloudFormation stack to another, create a cross-stack reference. Cross-stack references let you use a layered or service-oriented architecture. Instead of including all resources in a single stack, you create related AWS resources in separate stacks; then you can refer to required resource outputs from other stacks. By restricting cross-stack references to outputs, you control the parts of a stack that are referenced by other stacks.**

**For example, you might have a network stack with a VPC, a security group, and a subnet for public web applications, and a separate public web application stack. To ensure that the web applications use the security group and subnet from the network stack, you create a cross-stack reference that allows the web application stack to reference resource outputs from the network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.**

**To create a cross-stack reference, use the Export output field to flag the value of a resource output for export. Then, use the Fn::ImportValue intrinsic function to import the value. For more information, see** [**Outputs**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html) **and** [**Fn::ImportValue**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-importvalue.html)**.**

**ONE Lambda function is not working while we observed that other is consuming memory so how to fix it**

**Setup the reserved concurrency for second so that first gets it space**

**Different status from Application load balancer**

**HTTP 503 - HTTP 503 indicates 'Service unavailable' error. This error in ALB is an indicator of the target groups for the load balancer having no registered targets.**

**Incorrect options:**

**HTTP 500 - HTTP 500 indicates 'Internal server' error. There are several reasons for their error: A client submitted a request without an HTTP protocol, and the load balancer was unable to generate a redirect URL, there was an error executing the web ACL rules.**

**HTTP 504 - HTTP 504 is 'Gateway timeout' error. Several reasons for this error, to quote a few: The load balancer failed to establish a connection to the target before the connection timeout expired, The load balancer established a connection to the target but the target did not respond before the idle timeout period elapsed.**

**HTTP 403 - HTTP 403 is 'Forbidden' error. You configured an AWS WAF web access control list (web ACL) to monitor requests to your Application Load Balancer and it blocked a request.**

**IAM username and password - IAM username and password credentials cannot be used to access CodeCommit.**

**Incorrect options:**

**Git credentials - These are IAM -generated user name and password pair you can use to communicate with CodeCommit repositories over HTTPS.**

**SSH Keys - Are locally generated public-private key pair that you can associate with your IAM user to communicate with CodeCommit repositories over SSH.**

**AWS access keys - You can use these keys with the credential helper included with the AWS CLI to communicate with CodeCommit repositories over HTTPS.**

**Cloud Formation Template:-**

**!FindInMap [ MapName, TopLevelKey, SecondLevelKey ] - The intrinsic function Fn::FindInMap returns the value corresponding to keys in a two-level map that is declared in the Mappings section. YAML Syntax for the full function name: Fn::FindInMap: [ MapName, TopLevelKey, SecondLevelKey ]**

**Short form of the above syntax is : !FindInMap [ MapName, TopLevelKey, SecondLevelKey ]**

**Where,**

**MapName - Is the logical name of a mapping declared in the Mappings section that contains the keys and values. TopLevelKey - The top-level key name. Its value is a list of key-value pairs. SecondLevelKey - The second-level key name, which is set to one of the keys from the list assigned to TopLevelKey.**

**How to ssh into ec2 instances :**

**Key pairs - Key pairs consist of a public key and a private key. You use the private key to create a digital signature, and then AWS uses the corresponding public key to validate the signature. Key pairs are used only for Amazon EC2 and Amazon CloudFront. AWS does not provide key pairs for your account; you must create them. You can create Amazon EC2 key pairs from the Amazon EC2 console, CLI, or API. Key pairs make a robust combination for accessing an instance securely, a better option than using passwords.**

**Incorrect options:**

**Multi-Factor Authentication (MFA) - Multi-factor authentication (MFA) provides an extra level of security that you can apply to your AWS account. With MFA enabled, when you sign in to the AWS website, you are prompted for your user name and password, and an authentication code from an MFA device. Together, they provide increased security for your AWS account settings and resources. Its an added layer of protection for AWS account users.**

**Access keys - Access keys consist of two parts: an access key ID and a secret access key. You use access keys to sign programmatic requests that you make to AWS if you use AWS CLI commands (using the SDKs) or using AWS API operations. These credentials are for accessing AWS services programmatically and not for accessing the EC2 instance directly.**

**Root user credentials - Root user credentials are the Email ID and password used to create the AWS account. This user has full privileges on the account created and has access to all services under his account. The root user can create access keys or key pairs from his account. But, the root account credentials cannot directly be used to access EC2 instances or create digital signatures.**

**Secrets Manager(Allows Password store as wells as rotation)**

**AWS Secrets Manager enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to Secrets Manager APIs, eliminating the need to hardcode sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB.**

**SSM Parameter Store(Allows Password store) - AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. SSM Parameter Store cannot be used to automatically rotate the database credentials.**

**Systems Manager - AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services and allows you to automate operational tasks across your AWS resources. Systems Manager cannot be used to store your secrets securely and automatically rotate the database credentials.**

**KMS - AWS Key Management Service (KMS) makes it easy for you to create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications. KMS cannot be used to store your secrets securely and automatically rotate the database credentials.**

**To enable HTTPS connections for his web application deployed on the AWS Cloud, a developer is in the process of creating server certificate.**

**Which AWS entities can be used to deploy SSL/TLS server certificates?**

**AWS Certificate Manager (Provides SSH Inbuilt)- AWS Certificate Manager (ACM) is the preferred tool to provision, manage, and deploy server certificates. With ACM you can request a certificate or deploy an existing ACM or external certificate to AWS resources. Certificates provided by ACM are free and automatically renew. In a supported Region, you can use ACM to manage server certificates from the console or programmatically.**

**IAM (External certificate support it will provide ssh)- IAM is used as a certificate manager only when you must support HTTPS connections in a Region that is not supported by ACM. IAM securely encrypts your private keys and stores the encrypted version in IAM SSL certificate storage. IAM supports deploying server certificates in all Regions, but you must obtain your certificate from an external provider for use with AWS. You cannot upload an ACM certificate to IAM. Additionally, you cannot manage your certificates from the IAM Console.**

**Incorrect options:**

**AWS Secrets Manager - AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to Secrets Manager APIs, eliminating the need to hardcode sensitive information in plain text. It cannot be used to discover and protect your sensitive data in AWS.**

**AWS Systems Manager - AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services and allows you to automate operational tasks such as running commands, managing patches, and configuring servers across AWS Cloud as well as on-premises infrastructure.**

**AWS CloudFormation - AWS CloudFormation allows you to use programming languages or a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all Regions and accounts. Think infrastructure as code; think CloudFormation. You cannot use CloudFormation for running commands or managing patches on servers.**

**The team wants to analyze the incoming requests for latencies and the client's IP address patterns. → AWS Acesss Logs**

**Aws Request tracing provides only ip add pattern and not latency information**

**ALB access logs - Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues. Access logging is an optional feature of Elastic Load Balancing that is disabled by default**

**CloudTrail logs - Elastic Load Balancing is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Elastic Load Balancing. CloudTrail captures all API calls for Elastic Load Balancing as events. You can use AWS CloudTrail to capture detailed information about the calls made to the Elastic Load Balancing API and store them as log files in Amazon S3. You can use these CloudTrail logs to determine which API calls were made, the source IP address where the API call came from, who made the call, when the call was made, and so on.**

**CloudWatch metrics - Elastic Load Balancing publishes data points to Amazon CloudWatch for your load balancers and your targets. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as metrics. You can use metrics to verify that your system is performing as expected. This is the right feature if you wish to track a certain metric.**

**ALB request tracing - You can use request tracing to track HTTP requests. The load balancer adds a header with a trace identifier to each request it receives. Request tracing will not help you to analyze latency specific data.**

**Elastic Load balancer and its features to build highly available system across multiple azs**

**A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.**

**Elastic Load Balancing supports three types of load balancers:**

**Application Load Balancers**

**Network Load Balancers**

**Classic Load Balancers**

**Separate public traffic from private traffic - The nodes of an internet-facing load balancer have public IP addresses. Load balancers route requests to your targets using private IP addresses. Therefore, your targets do not need public IP addresses to receive requests from users over the internet.**

**Build a highly available system - Elastic Load Balancing provides fault tolerance for your applications by automatically balancing traffic across targets – Amazon EC2 instances, containers, IP addresses, and Lambda functions – in multiple Availability Zones while ensuring only healthy targets receive traffic.**

**Incorrect options:**

**The Load Balancer communicates with the underlying EC2 instances using their public IPs - This is an incorrect statement. The Load Balancer communicates with the underlying EC2 instances using their private IPs.**

**Improve vertical scalability of the system - This is an incorrect statement. Elastic Load Balancers can connect with Auto Scale groups to provide horizontal scaling.**

**Deploy EC2 instances across multiple AWS Regions - A Load Balancer can target EC2 instances only within an AWS Region.**

**Third Party Authorization for Lambda**

**"Lambda Authorizer"**

**An Amazon API Gateway Lambda authorizer (formerly known as a custom authorizer) is a Lambda function that you provide to control access to your API. A Lambda authorizer uses bearer token authentication strategies, such as OAuth or SAML. Before creating an API Gateway Lambda authorizer, you must first create the AWS Lambda function that implements the logic to authorize and, if necessary, to authenticate the caller.**

** via -** [**https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html**](https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html)

**Incorrect options:**

**"IAM permissions with sigv4" - Signature Version 4 is the process to add authentication information to AWS requests sent by HTTP. You will still need to provide permissions but our requirements have a need for 3rd party authentication which is where Lambda Authorizer comes in to play.**

**"Cognito User Pools" - A Cognito user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign-in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK. This is managed by AWS, therefore, does not meet our requirements.**

**"API Gateway User Pools" - This is a made-up option, added as a distractor.**

**"Lambda Authorizer"**

**An Amazon API Gateway Lambda authorizer (formerly known as a custom authorizer) is a Lambda function that you provide to control access to your API. A Lambda authorizer uses bearer token authentication strategies, such as OAuth or SAML. Before creating an API Gateway Lambda authorizer, you must first create the AWS Lambda function that implements the logic to authorize and, if necessary, to authenticate the caller.**

**via - https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html**

**Incorrect options:**

**"IAM permissions with sigv4" - Signature Version 4 is the process to add authentication information to AWS requests sent by HTTP. You will still need to provide permissions but our requirements have a needBurstable performance instances, which are T3, T3a, and T2 instances, are designed to provide a baseline level of CPU performance with the ability to burst to a higher level when required by your workload. Burstable performance instances are the only instance types that use credits for CPU usage.**

**0 seconds - AWS states that, if your AWS account is less than 12 months old, you can use a t2.micro instance for free within certain usage limits. for 3rd party authentication which is where Lambda Authorizer comes in to play.**

**"Cognito User Pools" - A Cognito user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign-in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK. This is managed by AWS, therefore, does not meet our requirements.**

**"API Gateway User Pools" - This is a made-up option, added as a distractor.**

**Burstable performance instances, which are T3, T3a, and T2 instances, are designed to provide a baseline level of CPU performance with the ability to burst to a higher level when required by your workload. Burstable performance instances are the only instance types that use credits for CPU usage.**

**0 seconds - AWS states that, if your AWS account is less than 12 months old, you can use a t2.micro instance for free within certain usage limits.**

**You manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. These policies grant the specified principal permission to perform specific actions on that resource and define under what conditions this applies.**

**Trust policy - Trust policies define which principal entities (accounts, users, roles, and federated users) can assume the role. An IAM role is both an identity and a resource that supports resource-based policies. For this reason, you must attach both a trust policy and an identity-based policy to an IAM role. The IAM service supports only one type of resource-based policy called a role trust policy, which is attached to an IAM role.**

**Incorrect options:**

**AWS Organizations Service Control Policies (SCP) - If you enable all features of AWS organization, then you can apply service control policies (SCPs) to any or all of your accounts. SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU). The SCP limits permissions for entities in member accounts, including each AWS account root user. An explicit deny in any of these policies overrides the allow.**

**Access control list (ACL) - Access control lists (ACLs) are service policies that allow you to control which principals in another account can access a resource. ACLs cannot be used to control access for a principal within the same account. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs.**

**Permissions boundary - AWS supports permissions boundaries for IAM entities (users or roles). A permissions boundary is an advanced feature for using a managed policy to set the maximum permissions that an identity-based policy can grant to an IAM entity. An entity's permissions boundary allows it to perform only the actions that are allowed by both its identity-based policies and its permissions boundaries.**

**A cybersecurity firm wants to run their applications on single-tenant hardware to meet security guidelines.**

**Which of the following is the MOST cost-effective way of isolating their Amazon EC2 instances to a single tenant?**

**Dedicated Instances - Dedicated Instances are Amazon EC2 instances that run in a virtual private cloud (VPC) on hardware that's dedicated to a single customer. Dedicated Instances that belong to different AWS accounts are physically isolated at a hardware level, even if those accounts are linked to a single-payer account. However, Dedicated Instances may share hardware with other instances from the same AWS account that are not Dedicated Instances.**

**A Dedicated Host is also a physical server that's dedicated for your use. With a Dedicated Host, you have visibility and control over how instances are placed on the server.**

**Spot Instances - A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Your Spot Instance runs whenever capacity is available and the maximum price per hour for your request exceeds the Spot price. Any instance present with unused capacity will be allocated. Even though this is cost-effective, it does not fulfill the single-tenant hardware requirement of the client and hence is not the correct option.**

**Dedicated Hosts - An Amazon EC2 Dedicated Host is a physical server with EC2 instance capacity fully dedicated to your use. Dedicated Hosts allow you to use your existing software licenses on EC2 instances. With a Dedicated Host, you have visibility and control over how instances are placed on the server. This option is costlier than the Dedicated Instance and hence is not the right choice for the current requirement.**

**On-Demand Instances - With On-Demand Instances, you pay for compute capacity by the second with no long-term commitments. You have full control over its lifecycle—you decide when to launch, stop, hibernate, start, reboot, or terminate it. Hardware isolation is not possible and on-demand has one of the costliest instance charges and hence is not the correct answer for current requirements.**

**As an AWS Certified Developer Associate, you are given a document written in YAML that represents the architecture of a serverless application. The first line of the document contains Transform: 'AWS::Serverless-2016-10-31'.**

**What does the Transform section in the document represent?**

* **​**
* **It represents a Lambda function definition**

**AWS CloudFormation template is a JSON- or YAML-formatted text file that describes your AWS infrastructure. Templates include several major sections. The "Resources" section is the only required section. The optional "Transform" section specifies one or more macros that AWS CloudFormation uses to process your template.**

**The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML.**

**Presence of 'Transform' section indicates it is a Serverless Application Model (SAM) template - The AWS::Serverless transform, which is a macro hosted by AWS CloudFormation, takes an entire template written in the AWS Serverless Application Model (AWS SAM) syntax and transforms and expands it into a compliant AWS CloudFormation template. So, presence of "Transform" section indicates, the document is a SAM template.**

**An IT company has a HealthCare application with data security requirements such that the encryption key must be stored in a custom application running on-premises. The company wants to offload the data storage as well as the encryption process to Amazon S3 but continue to use the existing encryption keys.**

**Which of the following S3 encryption options allows the company to leverage Amazon S3 for storing data with given constraints?**

**Server-Side Encryption with Customer-Provided Keys (SSE-C)**

**You have the following options for protecting data at rest in Amazon S3:**

**Server-Side Encryption – Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.**

**Client-Side Encryption – Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.**

**For the given use-case, the company wants to manage the encryption keys via its custom application and let S3 manage the encryption, therefore you must use Server-Side Encryption with Customer-Provided Keys (SSE-C).**

**Which of the following security credentials can only be created by the AWS Account root user**

**For Amazon CloudFront, you use key pairs to create signed URLs for private content, such as when you want to distribute restricted content that someone paid for.**

**CloudFront Key Pairs - IAM users can't create CloudFront key pairs. You must log in using root credentials to create key pairs.**

**A Developer has been entrusted with the job of securing certain S3 buckets that are shared by a large team of users. Last time, a bucket policy was changed, the bucket was erroneously available for everyone, outside the organization too.**

**Which feature/service will help the developer identify similar security issues with minimum effort?**

#### **Explanation**

**Correct option:**

**IAM Access Analyzer - AWS IAM Access Analyzer helps you identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity. This lets you identify unintended access to your resources and data, which is a security risk.**

**You can set the scope for the analyzer to an organization or an AWS account. This is your zone of trust. The analyzer scans all of the supported resources within your zone of trust. When Access Analyzer finds a policy that allows access to a resource from outside of your zone of trust, it generates an active finding.**

**Incorrect options:**

**Access Advisor feature on IAM console - To help identify the unused roles, IAM reports the last-used timestamp that represents when a role was last used to make an AWS request. Your security team can use this information to identify, analyze, and then confidently remove unused roles. This helps improve the security posture of your AWS environments. This does not provide information about non-IAM entities such as S3, hence it's not a correct choice here.**

**S3 Object Lock - S3 Object Lock enables you to store objects using a "Write Once Read Many" (WORM) model. S3 Object Lock can help prevent accidental or inappropriate deletion of data, it is not the right choice for the current scenario.**

**S3 Analytics - By using Amazon S3 analytics Storage Class Analysis you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. You cannot use S3 Analytics to identify unintended access to your S3 resources.**

**AWS Auto scaling group can have EC2 instances in multiple zones within a region.**

**Create a CNAME record**

**A CNAME record maps DNS queries for the name of the current record, such as acme.example.com, to another domain (example.com or example.net) or subdomain (acme.example.com or zenith.example.org).**

**CNAME records can be used to map one domain name to another. Although you should keep in mind that the DNS protocol does not allow you to create a CNAME record for the top node of a DNS namespace, also known as the zone apex. For example, if you register the DNS name example.com, the zone apex is example.com. You cannot create a CNAME record for example.com, but you can create CNAME records for www.example.com, newproduct.example.com, and so on.**

**Please review the major differences between CNAME and Alias Records:**

**Incorrect options:**

**Create an A record - Used to point a domain or subdomain to an IP address. 'A record' cannot be used to map one domain name to another.**

**Create a PTR record - A Pointer (PTR) record resolves an IP address to a fully-qualified domain name (FQDN) as an opposite to what A record does. PTR records are also called Reverse DNS records. 'PTR record' cannot be used to map one domain name to another.**

**Create an Alias Record - Alias records let you route traffic to selected AWS resources, such as CloudFront distributions and Amazon S3 buckets. They also let you route traffic from one record in a hosted zone to another record. 3rd party websites do not qualify for these as we have no control over those. 'Alias record' cannot be used to map one domain name to another.**

**You decide to make the APIs available to mobile developers as product offerings.**

**Use API Gateway Usage Plans**

**Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud.**

**Use AWS Billing Usage Plans - AWS Billing and Cost Management is the service that you use to pay your AWS bill, monitor your usage, and analyze and control your costs. There is no such thing as AWS Billing Usage Plans. You cannot use AWS Billing to set up public APIs for the application.**

**Use CloudFront Usage Plans - Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment. There is no such thing as CloudFront Usage Plans. You cannot use CloudFront to set up public APIs for the application.**

**Use AWS Lambda Custom Authorizers - Lambda is a separate service than Gateway API, therefore, it cannot be used to determine the API usage limits.**

**Access Advisor feature on IAM console- To help identify the unused roles, IAM reports the last-used timestamp that represents when a role was last used to make an AWS request. Your security team can use this information to identify, analyze, and then confidently remove unused roles. This helps improve the security posture of your AWS environments. Additionally, by removing unused roles, you can simplify your monitoring and auditing efforts by focusing only on roles that are in use.**

**Incorrect options:**

**AWS Trusted Advisor - AWS Trusted Advisor is an online tool that provides you real-time guidance to help you provision your resources following AWS best practices on cost optimization, security, fault tolerance, service limits, and performance improvement.**

**IAM Access Analyzer - AWS IAM Access Analyzer helps you identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity. This lets you identify unintended access to your resources and data, which is a security risk.**

**Amazon Inspector - Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for exposure, vulnerabilities, and deviations from best practices.**

**Cloud Formation Template**

**'Dependencies' section of the template - As you can see, there is no section called 'Dependencies' in the template. Although dependencies can be mentioned, there is no section itself for dependencies.**

**Incorrect options:**

**'Conditions' section of the template - This optional section includes conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.**

**'Resources' section of the template - This is the only required section and specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.**

**'Parameters' section of the template - This optional section is helpful in passing Values 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.**

**A firm runs its technology operations on a fleet of Amazon EC2 instances. The firm needs a certain software to be available on the instances to support their daily workflows. The developer team has been told to use the user data feature of EC2 instances.**

**User Data is generally used to perform common automated configuration tasks and even run scripts after the instance starts. When you launch an instance in Amazon EC2, you can pass two types of user data - shell scripts and cloud-init directives. You can also pass this data into the launch wizard as plain text or as a file.**

**By default, scripts entered as user data are executed with root user privileges - Scripts entered as user data are executed as the root user, hence do not need the sudo command in the script. Any files you create will be owned by root; if you need non-root users to have file access, you should modify the permissions accordingly in the script.**

**By default, user data runs only during the boot cycle when you first launch an instance - By default, user data scripts and cloud-init directives run only during the boot cycle when you first launch an instance. You can update your configuration to ensure that your user data scripts and cloud-init directives run every time you restart your instance.**

**Incorrect options:**

**By default, user data is executed every time an EC2 instance is re-started - As discussed above, this is not a default configuration of the system. But, can be achieved by explicitly configuring the instance.**

**When an instance is running, you can update user data by using root user credentials - You can't change the user data if the instance is running (even by using root user credentials), but you can view it.**

**By default, scripts entered as user data do not have root user privileges for executing - Scripts entered as user data are executed as the root user, hence do not need the sudo command in the script.**

**Your company has configured AWS Organizations to manage multiple AWS accounts. Within each AWS account, there are many CloudFormation scripts running. Your manager has requested that each script output the account number of the account the script was executed in.**

**Which Pseudo parameter will you use to get this information?s**

**Your company has configured AWS Organizations to manage multiple AWS accounts. Within each AWS account, there are many CloudFormation scripts running. Your manager has requested that each script output the account number of the account the script was executed in.**

**Which Pseudo parameter will you use to get this information?**

**AWS::AccountId**

**Using CloudFormation, you can create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you.**

**Pseudo parameters are parameters that are predefined by AWS CloudFormation. You do not declare them in your template. Use them the same way as you would a parameter, as the argument for the Ref function.**

**AWS::AccountId returns the AWS account ID of the account in which the stack is being created.**

**Incorrect options:**

**AWS::NoValue - This removes the corresponding resource property when specified as a return value in the Fn::If intrinsic function.**

**AWS::Region - Returns a string representing the AWS Region in which the encompassing resource is being created, such as us-west-2.**

**AWS::StackName - Returns the name of the stack as specified with the aws cloudformation create-stack command, such as "teststack".**