CATOR28

Lab :278Data Protection using Encryption

David Joseph

## Lab overview

Cryptography is the conversion of communicated information into secret code that keeps the information confidential and private. Functions include authentication, data integrity, and nonrepudiation. The central function of cryptography is *encryption*, which transforms data into an unreadable form.

Encryption ensures privacy by keeping the information hidden from people who the information is not intended for. *Decryption*, the opposite of encryption, transforms encrypted data back into data; it won’t make any sense until it has been properly decrypted.

In this lab, you will connect to a file server that is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance. You will configure the AWS Encryption command line interface (CLI) on the instance. You will create an encryption key by using the AWS Key Management Service (AWS KMS). The key will be used to encrypt and decrypt data. Next, you will create multiple text files that are unencrypted by default. You will then use the AWS KMS key to encrypt the files and view them while they are encrypted. You will finish the lab by decrypting the same files and viewing the contents.

**Objectives**

After completing this lab, you should be able to:

* Create an AWS KMS encryption key
* Install the AWS Encryption CLI
* Encrypt plaintext
* Decrypt ciphertext

**Accessing the AWS Management Console**

At the upper-right corner of these instructions, choose **Start Lab**

At the top of these instructions, choose the green circle next to **AWS**

This option opens the AWS Management Console in a new browser tab. The system automatically signs you in.

**Tip**: If a new browser tab does not open, a banner or icon at the top of your browser might indicate that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop-ups**.

If you see a dialog prompting you to switch to the new console home, choose **Switch to the new Console Home**.

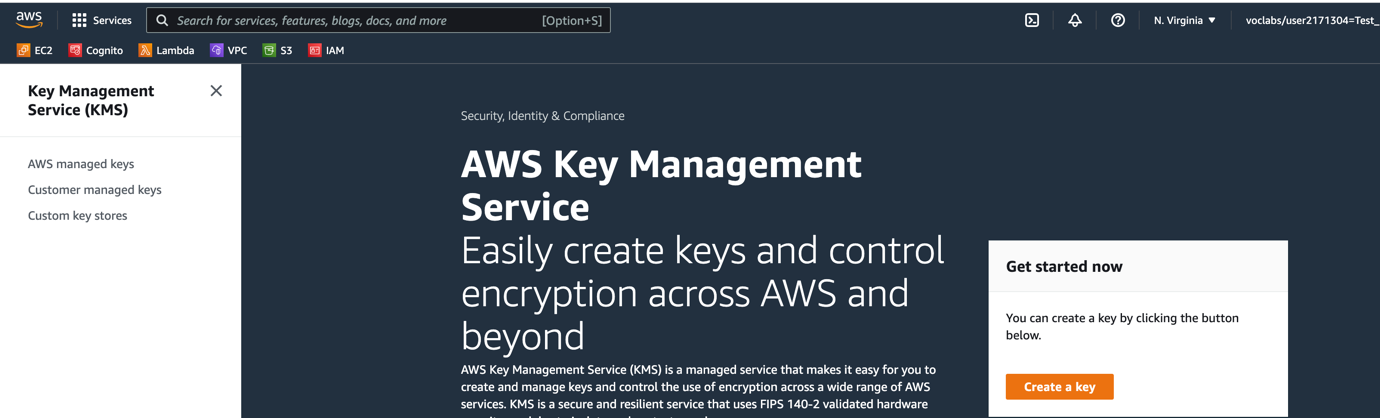
Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you should be able to see both browser tabs at the same time so that you can follow the lab steps.

**Task 1: Create an AWS KMS key**

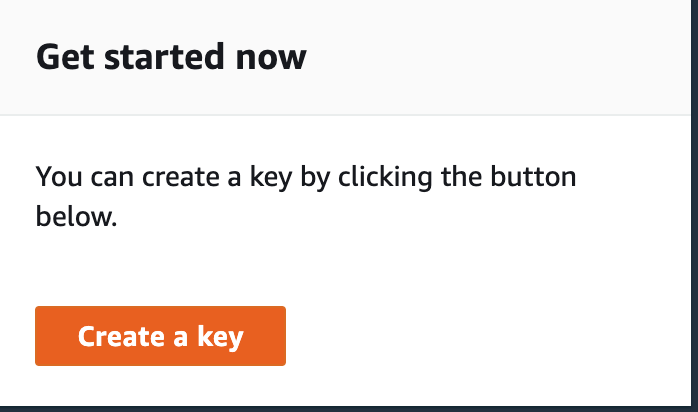
In this task, you will create an AWS KMS key that you will later use to encrypt and decrypt data.

With AWS KMS, you can create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications. AWS KMS is a secure and resilient service that uses hardware security modules (HSMs) that have been validated under the Federal Information Processing Standard (FIPS) Publication 140-2, or are in the process of being validated, to protect your keys.

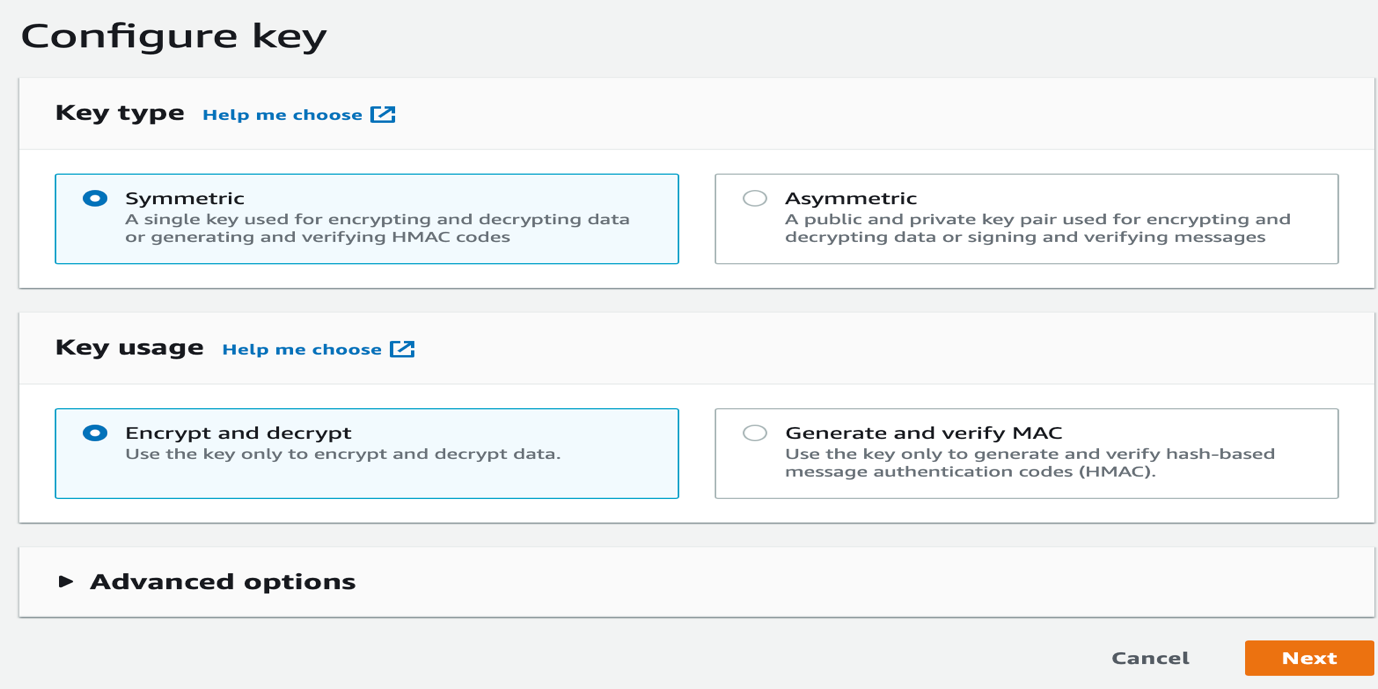
In the console, enter **KMS** in the search bar, and then choose **Key Management Service**.



Choose **Create a key**.

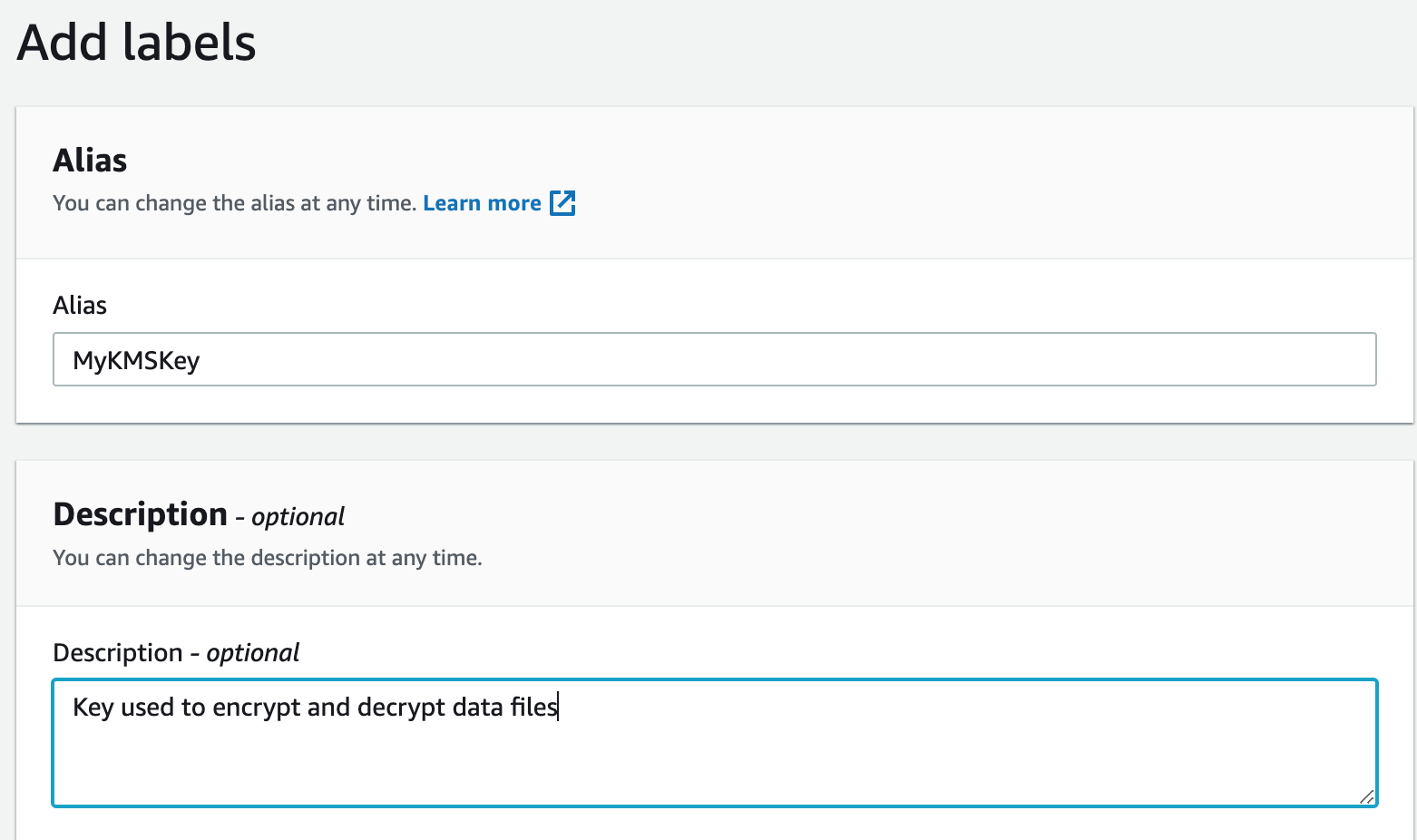


For **Key type**, choose **Symmetric**, and then choose **Next**. *Symmetric* encryption uses the same key to encrypt and decrypt data, which makes it fast and efficient to use. *Asymmetric* encryption uses a public key to encrypt data and a private key to decrypt information.



On the **Add labels** page, configure the following:

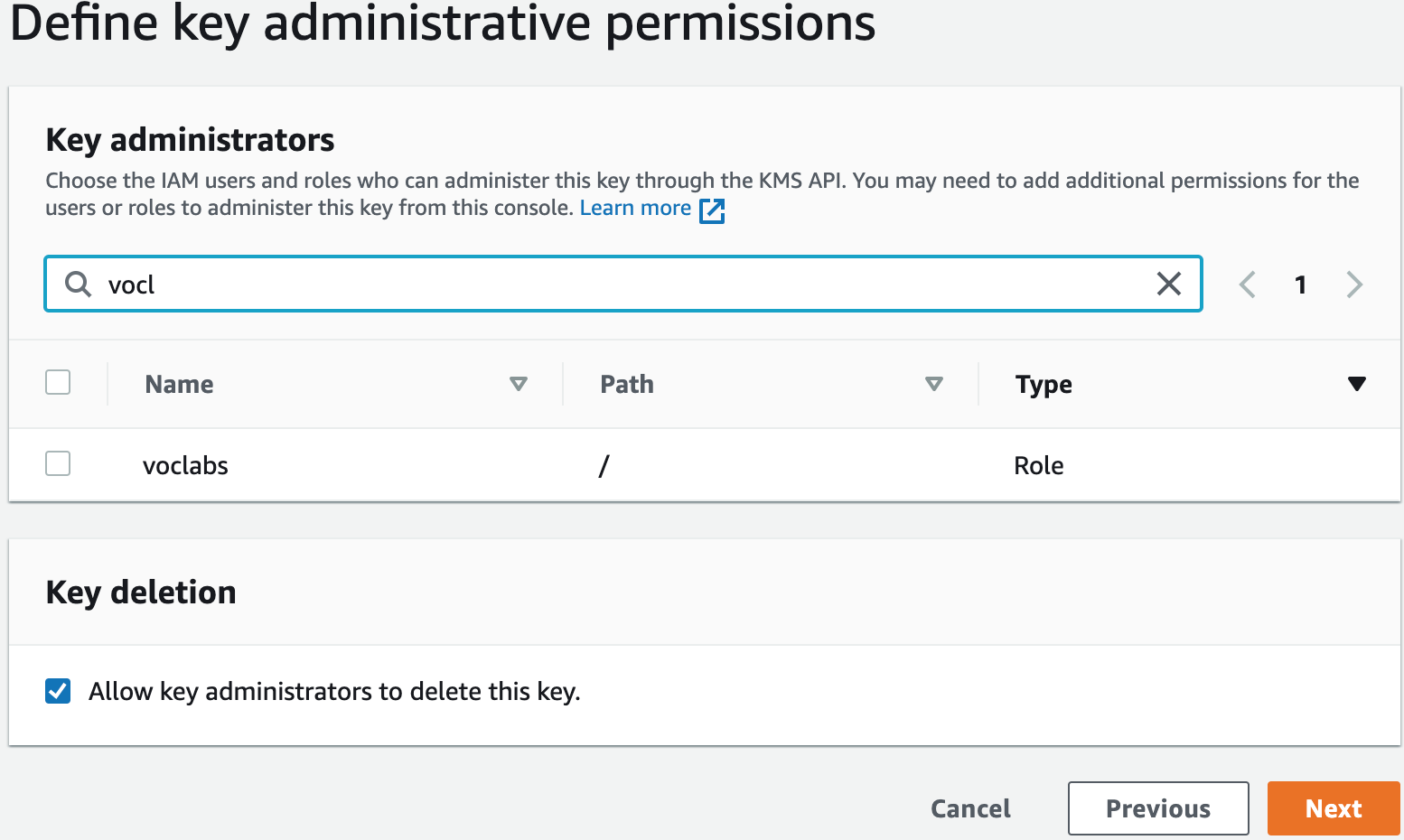
* + **Alias:** MyKMSKey
  + **Description:** Key used to encrypt and decrypt data files.

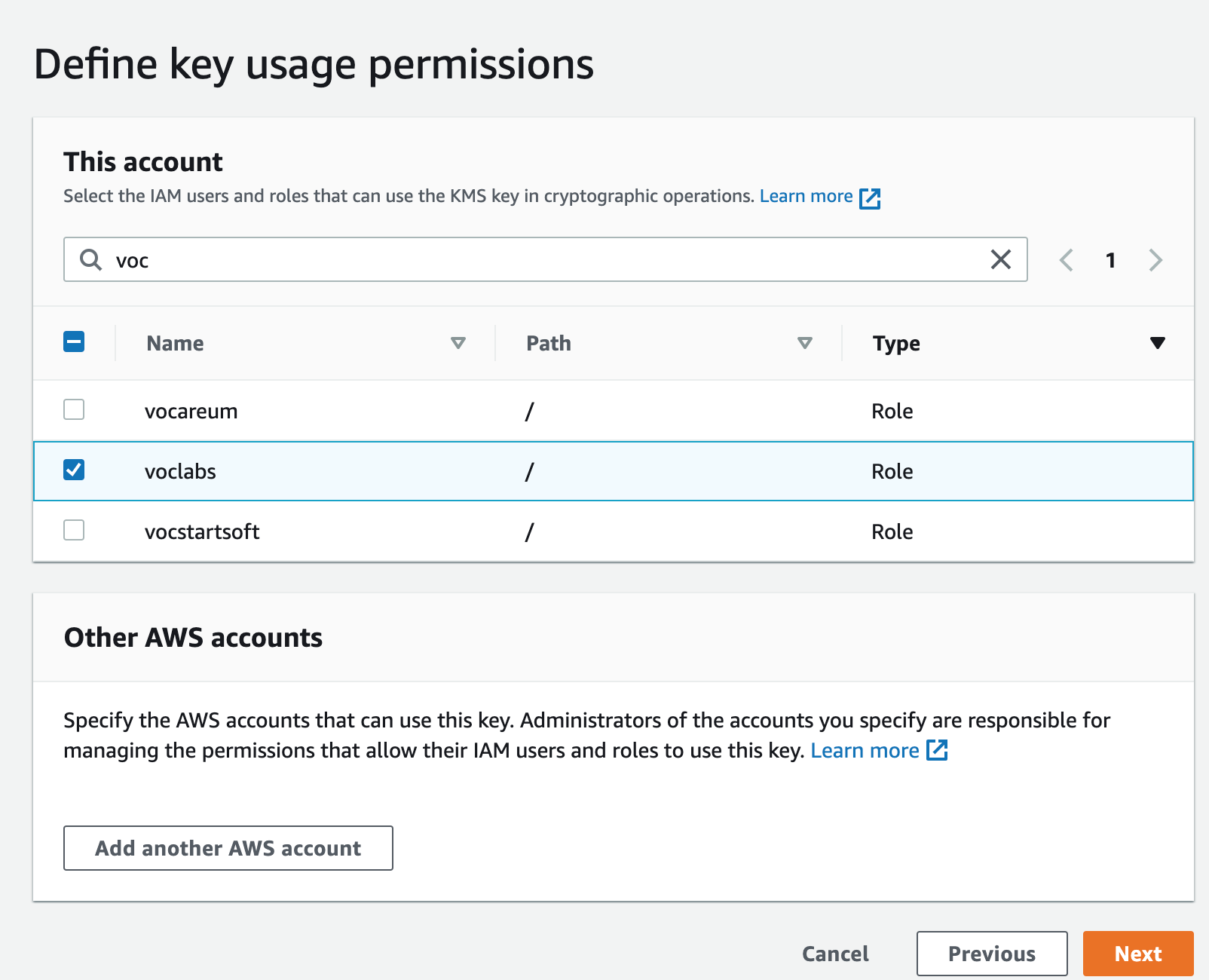


Choose **Next**.

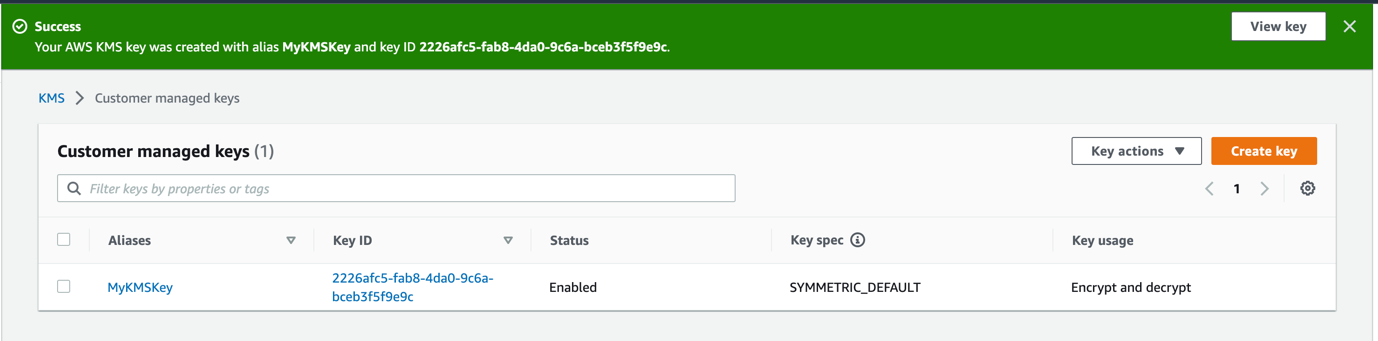
On the **Define key administrative permissions** page, in the **Key administrators** section, search for and select the check box for voclabs and then choose **Next**.

On the **Define key usage permissions** page, in the **This account** section, search for and select the check box for voclabs and then choose **Next**.

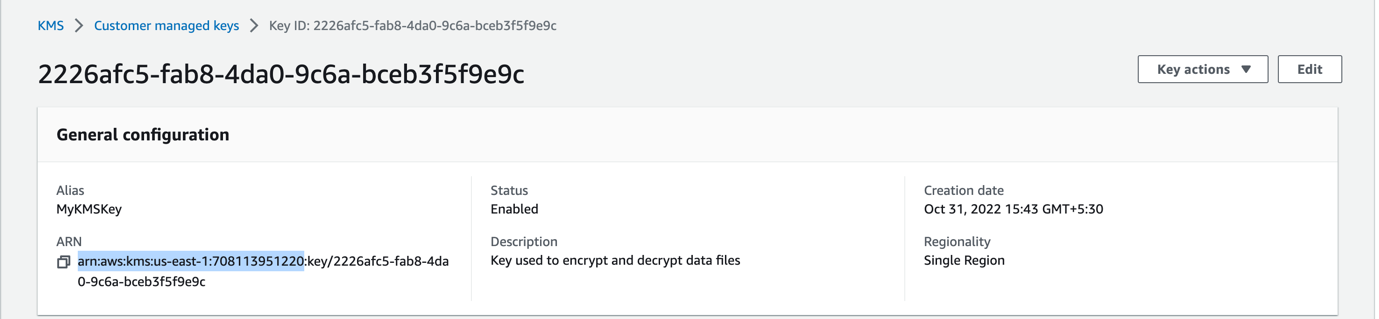




Review the settings, and then choose **Finish**.



Choose the link for **MyKMSKey**, which you just created, and copy the **ARN** (Amazon Resource Name) value to a text editor.



You will use this copied ARN later in the lab.

### Summary of task 1

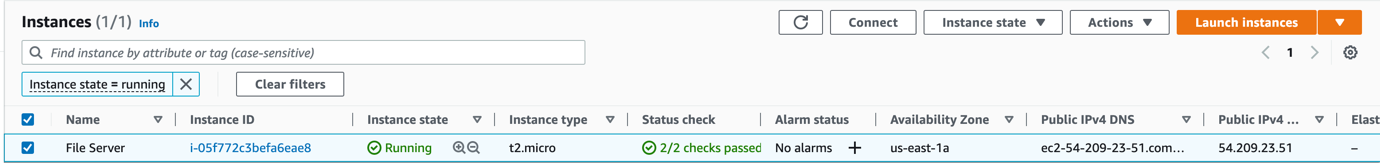
In this task, you created a symmetric AWS KMS key and gave ownership of that key to the **voclabs** IAM role that was pre-created for this lab.

## Task 2: Configure the File Server instance

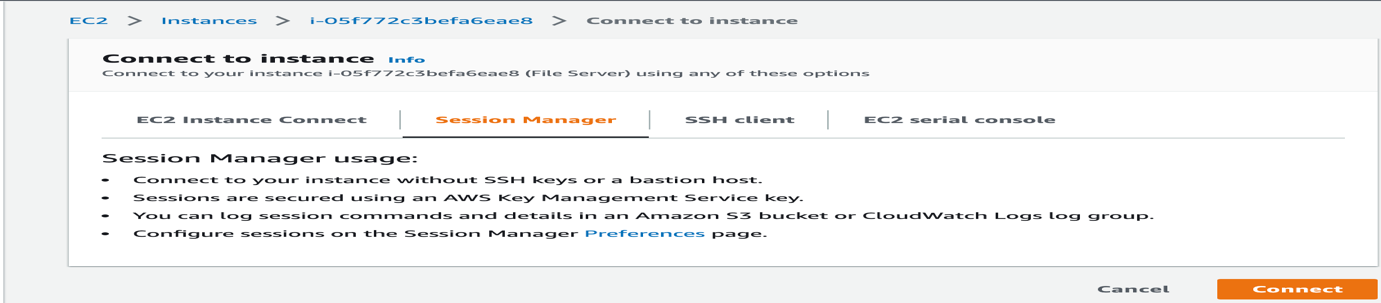
Before you can encrypt and decrypt data, you need to set up a few things. To use your AWS KMS key, you will configure AWS credentials on the File Server EC2 instance. After that, you will install the AWS Encryption CLI (aws-encryption-cli), which you can use to run encrypt and decrypt commands.

In the console, enter EC2 in the search bar, and then choose EC2.

In the Instances list, select the check box next for the File Server instance, and then choose Connect.



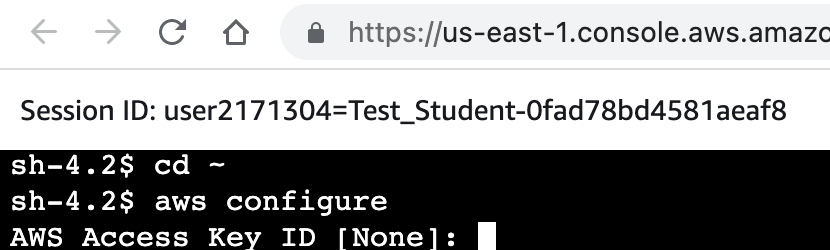
Choose the Session Manager tab, and then choose Connect.



To change to the home directory and create the AWS credentials file, run the following commands:

cd ~

aws configure



* **AWS Access Key ID**: Enter 1, and then press Enter.
* **AWS Secret Access Key**: Enter 1, and then press Enter.
* **Default region name**: Copy and paste the Region provided from the Vocareum **AWS Details** page.
* **Tip** You may need to press Ctrl+Shift+V to paste into Session Manager.
* **Default output format**: Press Enter.

The AWS configuration file is created, and you will update it in a later step. The previous entries of 1 are temporary placeholders.

Navigate to the Vocareum console page, and choose the **AWS Details** button.

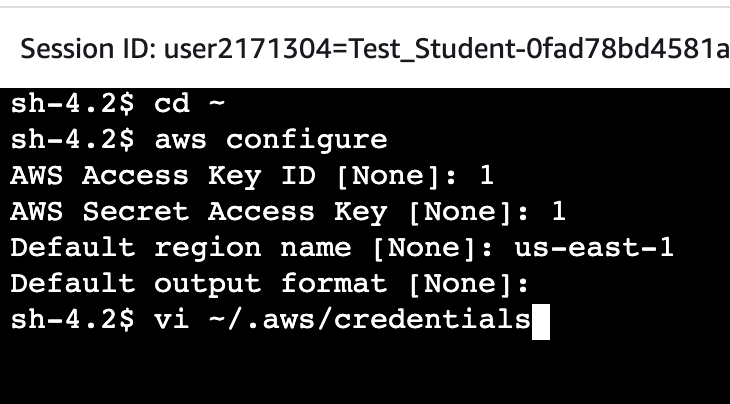
Next to **AWS CLI**, choose **Show**.

Copy and paste the code block, which starts with [default], into a text editor.

Return to the browser tab where you are logged in to the File Server.

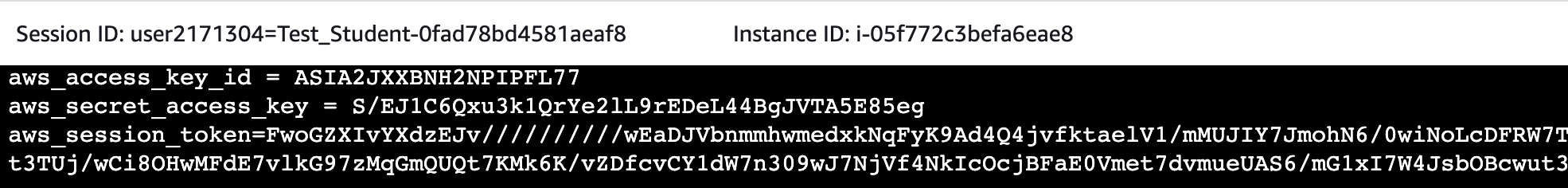
To open the AWS credentials file, run the following command:

vi ~/.aws/credentials



Paste in the code block that you copied from Vocareum.

The AWS credentials file should now look similar to the following:



To save and close the file, press Escape, type :wq and then press Enter.

To view the updated contents of the file, run the following command:

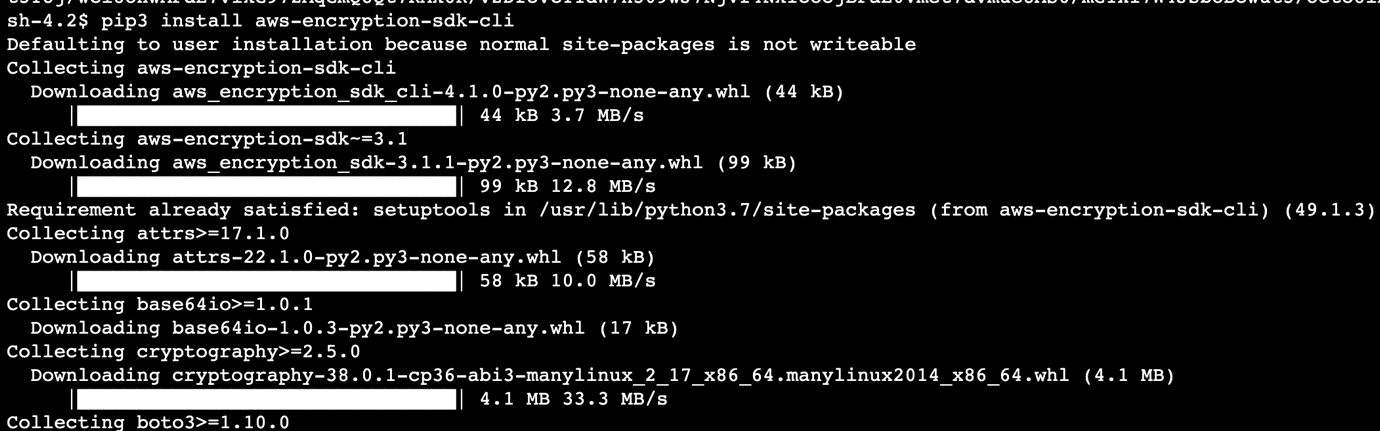
cat ~/.aws/credentials



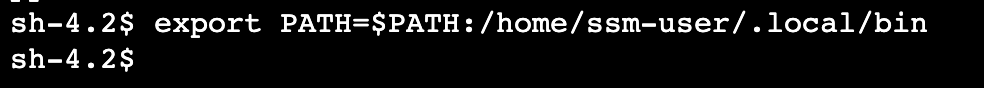
Now you will install the AWS Encryption CLI and export your path. By doing this, you will be able to run the commands to encrypt and decrypt data.

To install the AWS Encryption CLI and set your path, run the following commands:

pip3 install aws-encryption-sdk-cli



export PATH=$PATH:/home/ssm-user/.local/bin



### Summary of task 2

In this task, you configured the AWS credentials file, which provides the ability to use the AWS KMS key that you created earlier. You then installed the AWS Encryption CLI, so that you can run encryption commands.

## Task 3: Encrypt and decrypt data

In this task, you will create a text file with mock sensitive data in it. You will then use encryption to secure the file contents. Then, you will decrypt the data and view the file contents.

To create the text file, run the following commands:

touch secret1.txt secret2.txt secret3.txt

echo 'TOP SECRET 1!!!' > secret1.txt

To view the contents of the **secret1.txt** file, run the following command:

cat secret1.txt

One of the most important pillars of a well-architected framework is security. Thus, it is important to follow these **AWS security best practices** to prevent unnecessary security situations.

So, you’ve got a problem to solve and turned to AWS to build and host your solution. You create your account and now you’re all set up to brew some coffee and sit down at your workstation to architect, code, build, and deploy. Except, you aren’t.

[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-01.png)

There are many things **you must set up**if you want your solution to be **operative**, **secure**, **reliable**, **performant,** and **cost effective**. And, first things first, the best time to do that is now – right from the beginning, before you start to design and engineer.

## Initial AWS setup

Never, ever, use your root account for everyday use. Instead, head to Identity and Access Management (IAM) and create an administrator user. Protect and lock your root credentials in a secure place (is your password strong enough?) and, if your root user has keys generated, now is the best time to delete them.

You will absolutely want to **activate Multi Factor Authentication (MFA) too for your root account**. You must end up with a root user with MFA and no access keys. And you won’t use this user unless strictly necessary.

Now, about your newly created admin account, [activating MFA for it is a must](https://sysdig.com/blog/why-mfa-prevents-attacks/). It’s actually a requirement for every user in your account if you want to have a security first mindset (and you actually want to), but especially so for power users. You will only use this account for administrative purposes.

[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-02.png)

For daily use, you need to go to the IAM panel and create users, groups, and roles which can access only the resources to which you explicitly grant permissions.

Now you have:

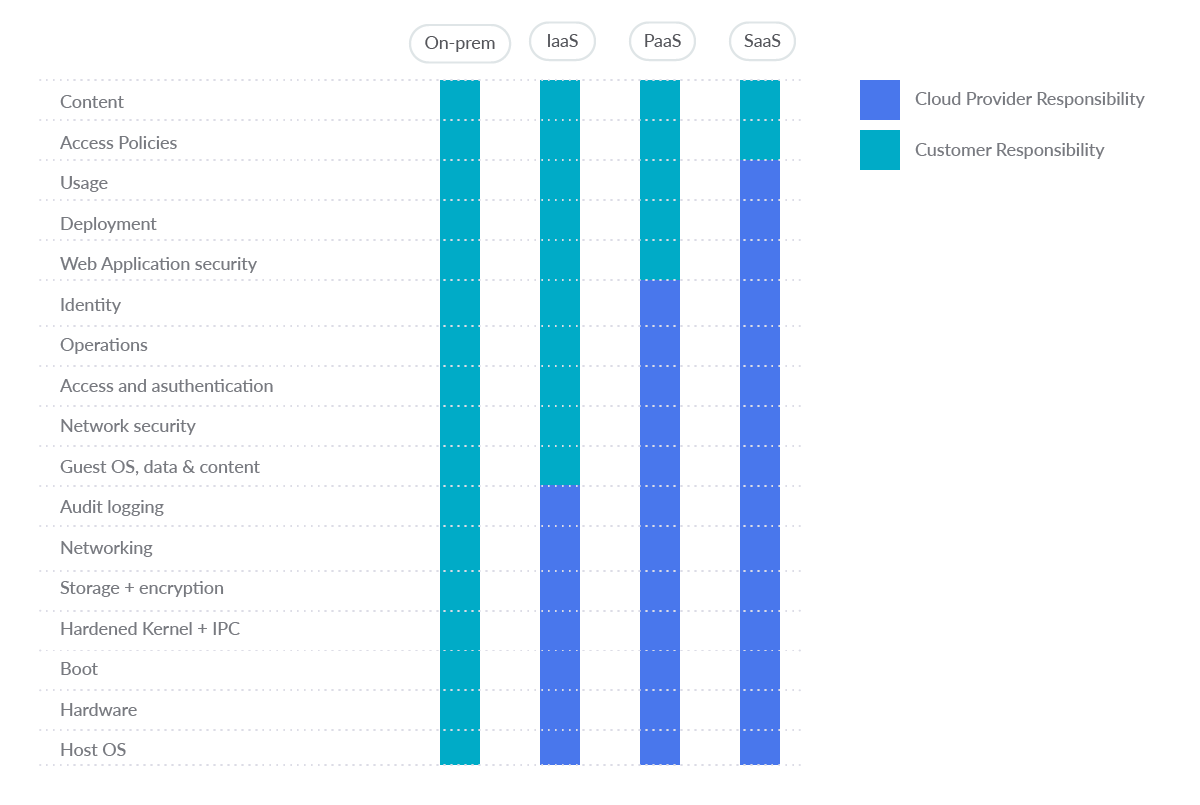
* Root account (with no keys) securely locked into a safe.
* Admin account for administrative use.
* Several users, groups, and roles for day to day use.

All of them should have MFA activated and strong passwords.

You are almost ready to begin your actual work, but first, a word of caution about the AWS shared responsibility model.

## AWS shared responsibility model

**Security and compliance is a shared responsibility between AWS and the customer**. AWS operates, manages, and controls the components from the host operating system and virtualization layer, down to the physical security of the facilities in which the service operates. The customer assumes responsibility and management of the guest operating system (including updates and security patches), other associated application software, as well as the configuration of the AWS provided [security group](https://sysdig.com/blog/aws-security-groups-guide/) firewall.

[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-03.png)

Therefore, the management and application of diligent AWS security is the responsibility of the customer.

## AWS Cloud security best practices checklist

In this section we will walk through the most common AWS services and provide**26 security best practices** to adopt.

**AWS Security best practices with open source**– [Cloud Custodian](https://cloudcustodian.io/) is a [Cloud Security Posture Management](https://sysdig.com/learn-cloud-native/cloud-security/cloud-security-posture-management/) (CSPM) tool. [CSPM tools](https://sysdig.com/products/secure/cspm-cloud-security-posture-management/) evaluate your cloud configuration and identify common configuration mistakes. They also monitor cloud logs to detect threats and configuration changes.

Now let’s walk through service by service.

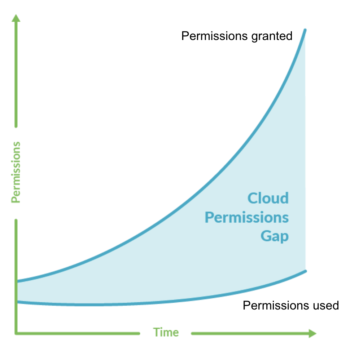
|  |  |  |  |
| --- | --- | --- | --- |
| **AWS security best practices by service** | | | |
|  | High Risk 🟥🟥🟥 | Medium Risk 🟨🟨 | Low Risk 🟩 |
| [AWS IAM](https://sysdig.com/blog/26-aws-security-best-practices/#IAM) | [(1)](https://sysdig.com/blog/26-aws-security-best-practices/#1) IAM policies should not allow full “\*” administrative privileges  [(4)](https://sysdig.com/blog/26-aws-security-best-practices/#4) IAM root user access key should not exist  [(6)](https://sysdig.com/blog/26-aws-security-best-practices/#6) Hardware MFA should be enabled for the root user | [(3)](https://sysdig.com/blog/26-aws-security-best-practices/#3) IAM users’ access keys should be rotated every 90 days or less  [(5)](https://sysdig.com/blog/26-aws-security-best-practices/#5) MFA should be enabled for all IAM users that have a console password  [(7)](https://sysdig.com/blog/26-aws-security-best-practices/#7) Password policies for IAM users should have strong configurations  [(8)](https://sysdig.com/blog/26-aws-security-best-practices/#8) Unused IAM user credentials should be removed | [(2)](https://sysdig.com/blog/26-aws-security-best-practices/#2) IAM users should not have IAM policies attached |
| [Amazon S3](https://sysdig.com/blog/26-aws-security-best-practices/#s3) | [(10)](https://sysdig.com/blog/26-aws-security-best-practices/#10) S3 buckets should have server-side encryption enabled | [(9)](https://sysdig.com/blog/26-aws-security-best-practices/#9) S3 Block Public Access setting should be enabled  [(11)](https://sysdig.com/blog/26-aws-security-best-practices/#11) S3 Block Public Access setting should be enabled at the bucket level |  |
| [AWS CloudTrail](https://sysdig.com/blog/26-aws-security-best-practices/#cloudtrail) | [(12)](https://sysdig.com/blog/26-aws-security-best-practices/#12) CloudTrail should be enabled and configured with at least one multi-Region trail | [(13)](https://sysdig.com/blog/26-aws-security-best-practices/#13) CloudTrail should have encryption at rest enabled  [(14)](https://sysdig.com/blog/26-aws-security-best-practices/#14) Ensure CloudTrail log file validation is enabled |  |
| [AWS Config](https://sysdig.com/blog/26-aws-security-best-practices/#config) | [(15)](https://sysdig.com/blog/26-aws-security-best-practices/#15) AWS Config should be enabled |  |  |
| [Amazon EC2](https://sysdig.com/blog/26-aws-security-best-practices/#ec2) | [(16)](https://sysdig.com/blog/26-aws-security-best-practices/#16) Attached EBS volumes should be encrypted at rest  [(19)](https://sysdig.com/blog/26-aws-security-best-practices/#19) EBS default encryption should be enabled |  | [(17)](https://sysdig.com/blog/26-aws-security-best-practices/#17) VPC flow logging should be enabled in all VPCs  [(18)](https://sysdig.com/blog/26-aws-security-best-practices/#18) The VPC default [security group](https://sysdig.com/blog/aws-security-groups-guide/) should not allow inbound and outbound traffic |
| [AWS DMS](https://sysdig.com/blog/26-aws-security-best-practices/#DMS) | [(20)](https://sysdig.com/blog/26-aws-security-best-practices/#20) AWS Database Migration Service replication instances should not be public |  |  |
| [Amazon EBS](https://sysdig.com/blog/26-aws-security-best-practices/#ebs) | [(21)](https://sysdig.com/blog/26-aws-security-best-practices/#21) Amazon EBS snapshots should not be public, determined by the ability to be restorable by anyone |  |  |
| [Amazon OpenSearch Service](https://sysdig.com/blog/26-aws-security-best-practices/#opensearch) | [(22)](https://sysdig.com/blog/26-aws-security-best-practices/#22) Elasticsearch domains should have encryption at rest enabled |  |  |
| [Amazon SageMaker](https://sysdig.com/blog/26-aws-security-best-practices/#SageMaker) |  | [(23)](https://sysdig.com/blog/26-aws-security-best-practices/#23) SageMaker notebook instances should not have direct internet access |  |
| [AWS Lambda](https://sysdig.com/blog/26-aws-security-best-practices/#Lambda) |  | [(24)](https://sysdig.com/blog/26-aws-security-best-practices/#24) Lambda functions should use supported runtimes |  |
| [AWS KMS](https://sysdig.com/blog/26-aws-security-best-practices/#KMS) |  | [(25)](https://sysdig.com/blog/26-aws-security-best-practices/#25) AWS KMS keys should not be unintentionally deleted |  |
| [Amazon GuardDuty](https://sysdig.com/blog/26-aws-security-best-practices/#guarduty) |  | [(26)](https://sysdig.com/blog/26-aws-security-best-practices/#26) GuardDuty should be enabled |  |

## AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) helps enforce least privilege access control to AWS resources. You can use IAM to restrict who is authenticated (signed in) and authorized (has permissions) to use resources.

### 1.- Do not allow full “\*” administrative privileges on IAM policies 🟥🟥🟥

IAM policies define a set of privileges that are granted to users, groups, or roles. Following standard security advice, **you should grant least privilege**, which means to allow only the permissions that are required to perform a task.

[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-04.png)

When you provide full administrative privileges instead of the minimum set of permissions that the user needs, [you expose the resources to potentially unwanted actions](https://sysdig.com/blog/threat-detection-aws-cloud-containers/).

For each AWS account, list the customer managed policies available:

aws iam list-policies --scope Local --query 'Policies[\*].Arn'

The previous command will return a list of policies along with their Amazon Resource Names (ARNs). Using these ARNs, now retrieve the policy document in JSON format:

aws iam get-policy-version

--policy-arn POLICY\_ARN

--version-id v1

--query 'PolicyVersion.Document'

The output should be the requested IAM policy document:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "1234567890",

"Effect": "Allow",

"Action": "\*",

"Resource": "\*"

}

]

}

Look into this document for the following elements:

"Effect": "Allow", "Action": "\*", "Resource": "\*"

If these elements are present, then the **customer-managed policy allows full administrative privileges**. **This is a risk and must be avoided**, so you will need to tune these policies down to pinpoint exactly what actions you want to allow for each specific resource.

Repeat the previous procedure for the other IAM customer managed policies.

If you want to detect the use of full administrative privileges with open source, here is a Cloud Custodian rule:

- name: full-administrative-privileges

description: IAM policies are the means by which privileges are granted to users, groups, or roles. It is recommended and considered a standard security advice to grant least privilege -that is, granting only the permissions required to perform a task. Determine what users need to do and then craft policies for them that let the users perform only those tasks, instead of allowing full administrative privileges.

resource: iam-policy

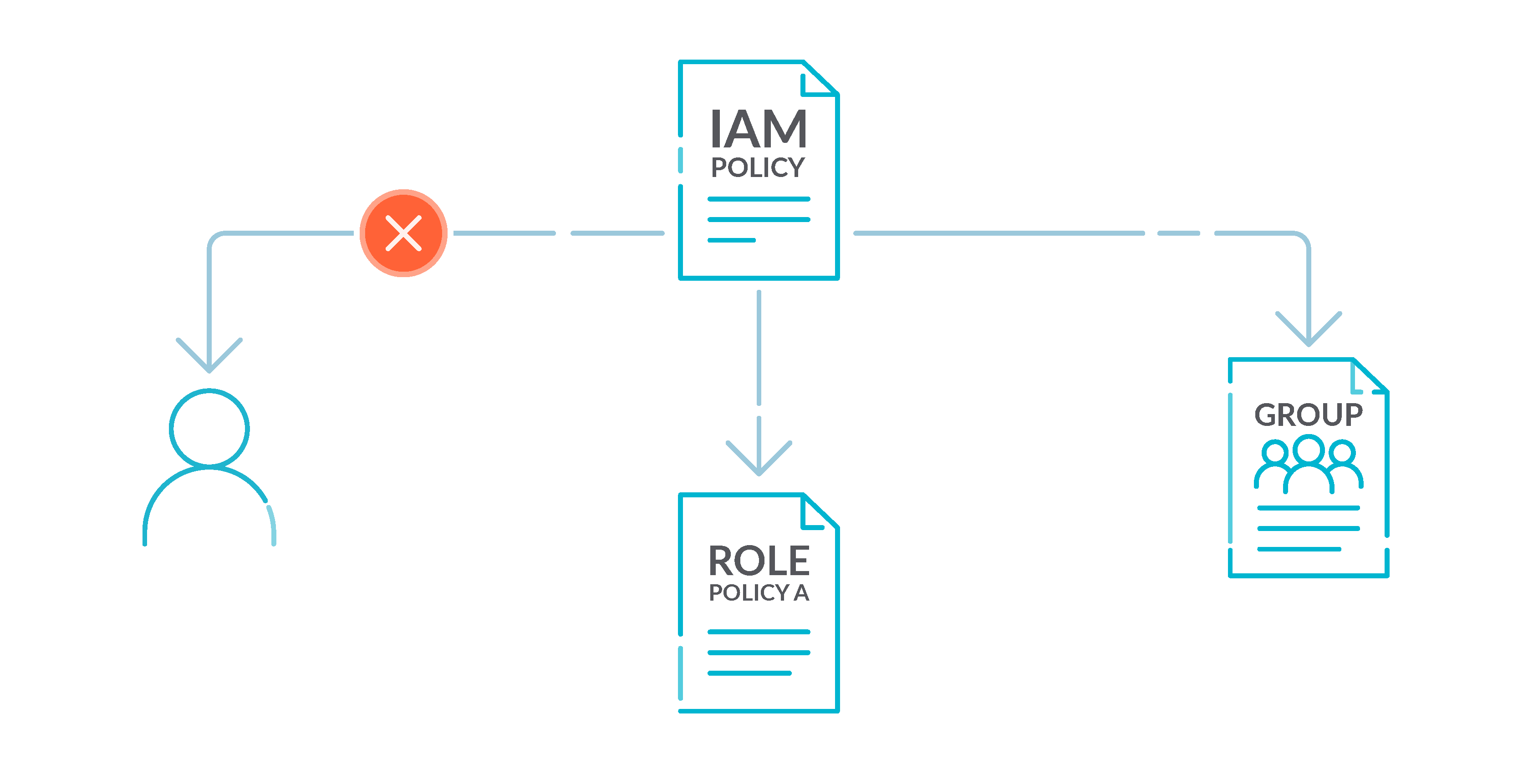
filters:

- type: used

- type: has-allow-all

### 2.- Do not attach IAM policies to users 🟩

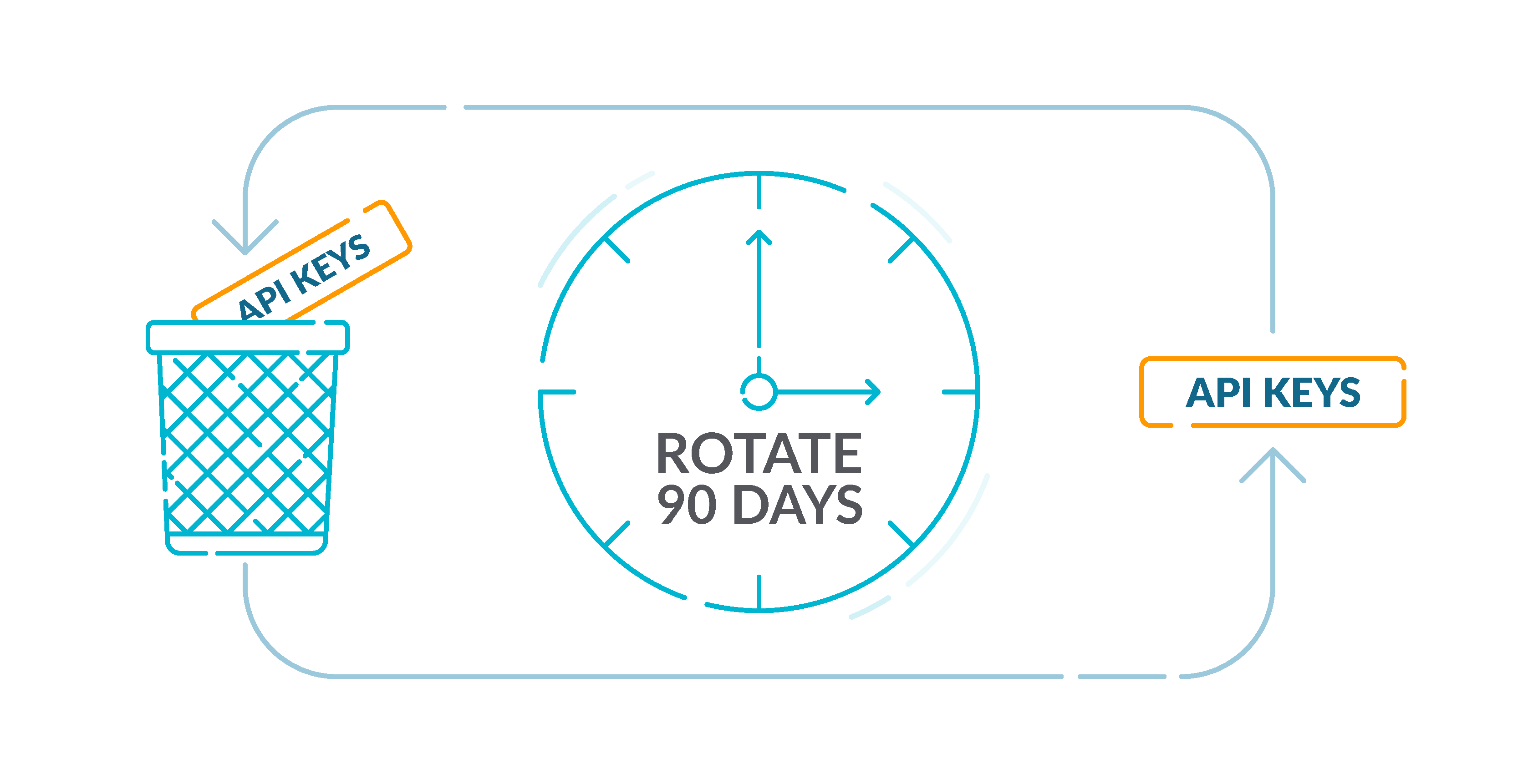
By default, IAM users, groups, and roles have no access to AWS resources.

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-policy.png)

IAM policies grant privileges to users, groups, or roles. We recommend that you **apply IAM policies directly to groups and roles** but not to users. Assigning privileges at the group or role level reduces the complexity of access management as the number of users grows. Reducing access management complexity might in turn reduce the opportunity for a principal to inadvertently receive or retain excessive privileges.

### 3.- Rotate IAM users’ access keys every 90 days or less 🟨🟨

[AWS recommends](https://docs.aws.amazon.com/general/latest/gr/aws-access-keys-best-practices.html) that you rotate the access keys every 90 days. Rotating access keys reduces the chance that an access key that is associated with a compromised or terminated account is used. It also ensures that data cannot be accessed with an old key that might have been lost, cracked, or stolen. Always update your applications after you rotate access keys.

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-rotate.png)

First, list all IAM users available in your AWS account with:

aws iam list-users --query 'Users[\*].UserName'

For all the users returned by this command, determine each active access key lifetime by doing:

aws iam list-access-keys --user-name USER\_NAME

This should expose the metadata for each access key existing for the specified IAM user. The output will look like this:

{

"AccessKeyMetadata": [

{

"UserName": "some-user",

"Status": "Inactive",

"CreateDate": "2022-05-18T13:43:23Z",

"AccessKeyId": "AAAABBBBCCCCDDDDEEEE"

},

{

"UserName": "some-user",

"Status": "Active",

"CreateDate": "2022-03-21T09:12:32Z",

"AccessKeyId": "AAAABBBBCCCCDDDDEEEE"

}

]

}

Check the CreateDate parameter value for each active key to determine its creation time. If an active access key has been created before the last 90 days, the key is outdated and must be [rotated](https://aws.amazon.com/blogs/security/how-to-rotate-access-keys-for-iam-users/) to secure the access to your AWS resources.

Repeat for each IAM user existing in your AWS account.

### 4.- Ensure that IAM root user access keys do not exist 🟥🟥🟥

As we stated during your initial setup, **we highly recommend that you remove all access keys that are associated with the root user**. This limits the vectors that can be used to compromise your account. It also encourages the creation and use of role-based accounts that are least privileged.

The following Cloud Custodian rule will check if root access keys have been used on your account:

- name: root-access-keys

description: The root user account is the most privileged user in an AWS account. AWS Access Keys provide programmatic access to a given AWS account. It is recommended that all access keys associated with the root user account be removed.

resource: account

filters:

- type: iam-summary

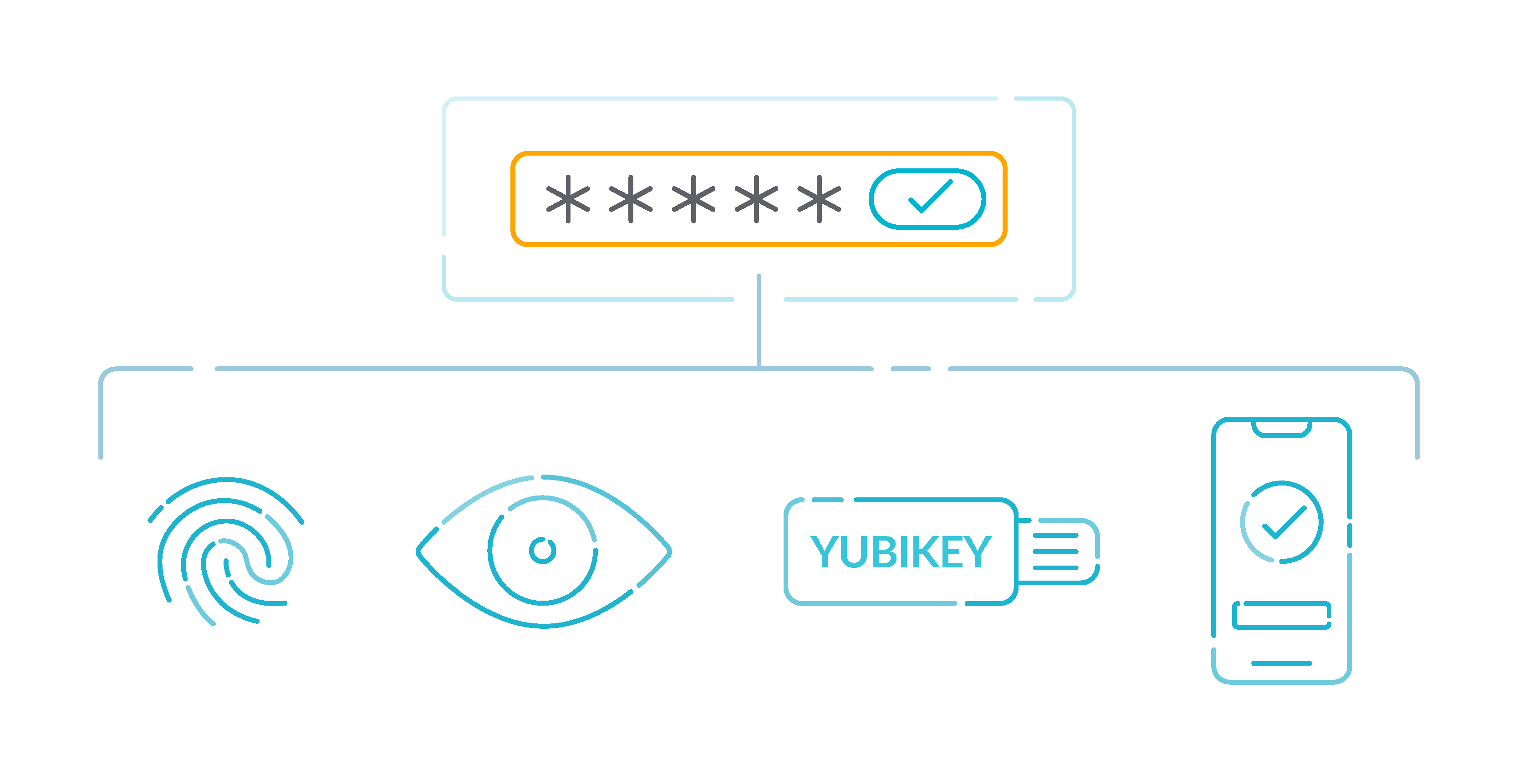
key: AccountAccessKeysPresent

value: 0

op: gt

### 5.- Enable MFA for all IAM users that have a console password 🟨🟨

Multi-factor authentication (MFA) adds an extra layer of protection on top of a username and password. With MFA enabled, when a user signs in to an AWS website, they are prompted for their username and password. In addition, they are prompted for an authentication code from their AWS MFA device.

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-mfa.png)

**We recommend that you**[**enable MFA for all accounts**](https://sysdig.com/blog/why-mfa-prevents-attacks/)**that have a console password**. MFA is designed to provide increased security for console access. The authenticating principal must possess a device that emits a time-sensitive key and must have knowledge of a credential.

### 6.- Enable hardware MFA for the root user 🟥🟥🟥

**Virtual MFA might not provide the same level of security as hardware MFA devices**. A hardware MFA has a minimal attack surface, and cannot be stolen unless the malicious user gains physical access to the hardware device. We recommend that you use only a virtual MFA device while you wait for hardware purchase approval or for your hardware to arrive, especially for root users.

To learn more, see [Enabling a virtual multi-factor authentication (MFA) device (console)](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html) in the IAM User Guide.

Here is a Cloud Custodian rule to detect lack of root hardware MFA:

- name: root-hardware-mfa

description: The root user account is the most privileged user in an AWS account. MFA adds an extra layer of protection on top of a username and password. With MFA enabled, when a user signs in to an AWS website, they will be prompted for their username and password as well as for an authentication code from their AWS MFA device. It is recommended that the root user account be protected with a hardware MFA.

resource: account

filters:

- or:

- type: iam-summary

key: AccountMFAEnabled

value: 1

op: ne

- and:

- type: iam-summary

key: AccountMFAEnabled

value: 1

op: eq

- type: has-virtual-mfa

value: true

### 7.- Ensure those password policies for IAM users have strong configurations 🟨🟨

We recommend that you enforce the creation of strong user passwords. You can set a **password policy** on your AWS account to **specify complexity requirements** and **mandatory rotation periods** for passwords.

When you create or change a password policy, most of the password policy settings are enforced the next time users change their passwords. Some of the settings are enforced immediately.

What constitutes a strong password is a subjective matter, but the following settings will put you on the right path:

RequireUppercaseCharacters: true

RequireLowercaseCharacters: true

RequireSymbols: true

RequireNumbers: true

MinimumPasswordLength: 8

### 8.- Remove unused IAM user credentials 🟨🟨

IAM users can access AWS resources using different types of credentials, such as passwords or access keys. We recommend you remove or deactivate all credentials that were unused for 90 days or more to reduce the window of opportunity for credentials associated with a compromised or abandoned account to be used.

You can use the IAM console to get some of the information that you need to monitor accounts for dated credentials. For example, when you view users in your account, there is a column for Access key age, Password age, and Last activity. If the value in any of these columns is greater than 90 days, make the credentials for those users inactive.

You can also use credential reports to monitor user accounts and identify those with no activity for 90 or more days. You can download credential reports in .csv format from the IAM console.

For more information, check out [AWS security best practices for IAM](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html) in more detail.

## Amazon S3

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. There are [few AWS security best practices to adopt when it comes to S3](https://aws.amazon.com/blogs/security/top-10-security-best-practices-for-securing-data-in-amazon-s3/).

### 9.- Enable S3 Block Public Access setting 🟨🟨

Amazon S3 public access block is designed to provide controls across an entire AWS account or at the individual S3 bucket level to ensure that objects never have public access. Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both.

[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-s3-exposed.png)

Unless you intend to have your S3 buckets be publicly accessible, you should configure the account level Amazon S3 Block Public Access feature.

Get the names of all S3 buckets available in your AWS account:

aws s3api list-buckets --query 'Buckets[\*].Name'

For each bucket returned, get its S3 Block Public Access feature configuration:

aws s3api get-public-access-block --bucket BUCKET\_NAME

The output for the previous command should be like this:

"PublicAccessBlockConfiguration": {

"BlockPublicAcls": false,

"IgnorePublicAcls": false,

"BlockPublicPolicy": false,

"RestrictPublicBuckets": false

}

**If any of these values is false, then your data privacy is at stake**. Use this short command to remediate it:

aws s3api put-public-access-block

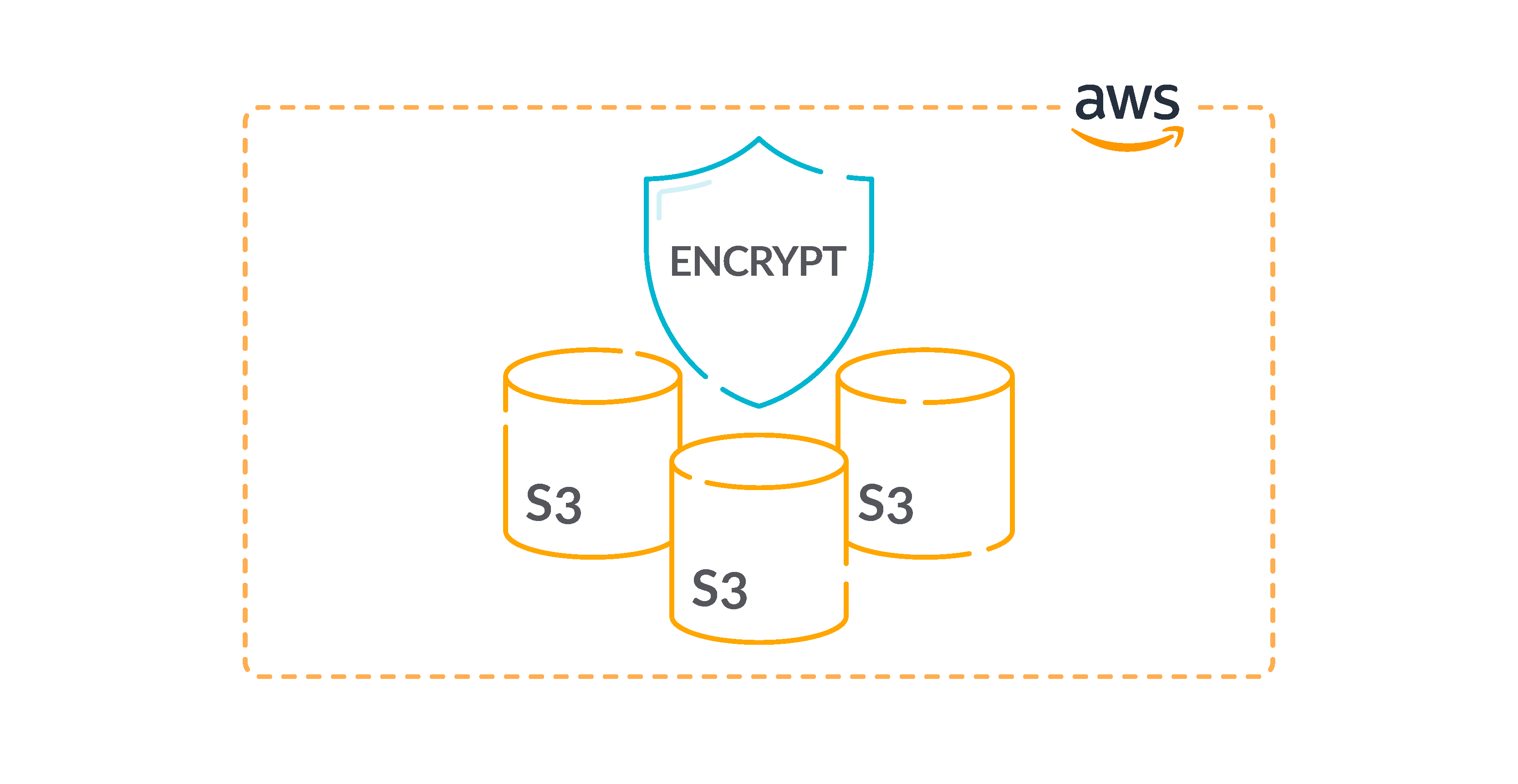
--region REGION

--bucket BUCKET\_NAME

--public-access-block-configuration BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true

### 10.- Enable server-side encryption on S3 buckets 🟥🟥🟥

For an added layer of security for your sensitive data in S3 buckets, **you should configure your buckets with server-side encryption to protect your data at rest**. Amazon S3 encrypts each object with a unique key. As an additional safeguard, Amazon S3 encrypts the key itself with a root key that it rotates regularly. Amazon S3 server-side encryption uses one of the strongest block ciphers available to encrypt your data, 256-bit Advanced Encryption Standard (AES-256).

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-s3.png)

List all existing S3 buckets available in your AWS account:

aws s3api list-buckets --query 'Buckets[\*].Name'

Now, use the names of the S3 buckets returned at the previous step as identifiers to retrieve their Default Encryption feature status:

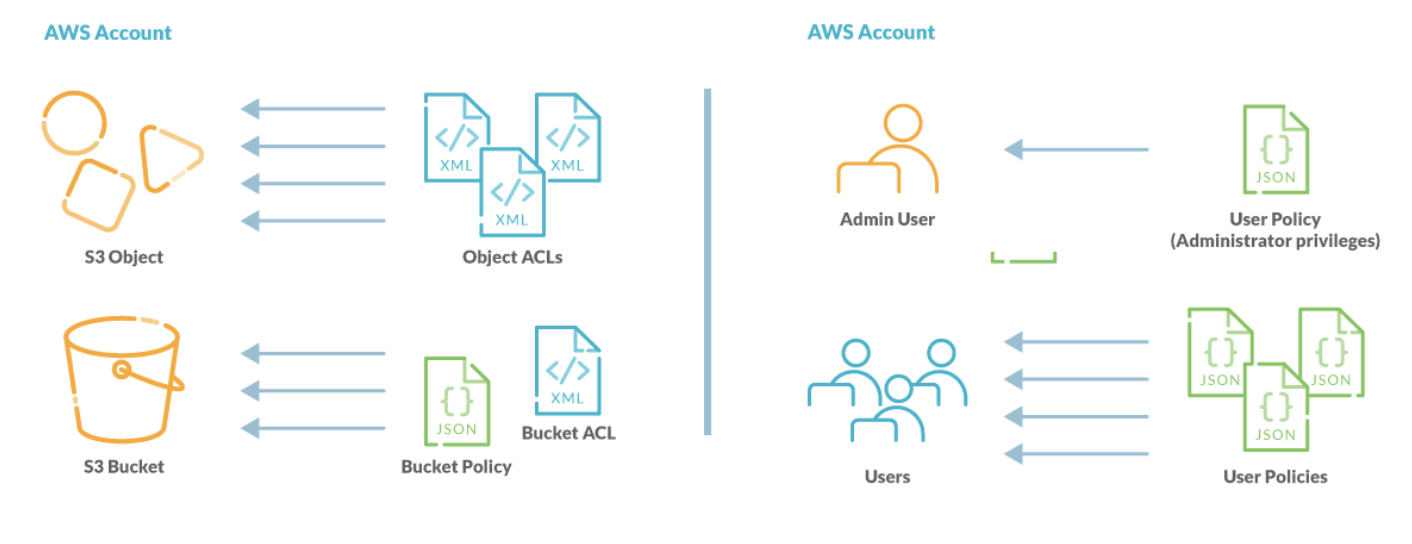
aws s3api get-bucket-encryption --bucket BUCKET\_NAME

The command output should return the requested feature configuration details. If the get-bucket-encryption command output returns an error message, the default encryption is not currently enabled, and therefore the selected S3 bucket does not automatically encrypt all objects when stored in Amazon S3.

Repeat this procedure for all your S3 buckets.

### 11.- Enable S3 Block Public Access setting at the bucket level 🟨🟨

Amazon S3 public access block is designed to provide controls across an entire AWS account or at the individual S3 bucket level to ensure that objects never have public access. Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both.

​​[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-s3-policy.png)

Unless you intend to have your S3 buckets be publicly accessible, which you probably shouldn’t, you should configure the account level Amazon S3 Block Public Access feature.

You can use this Cloud Custodian rule to detect S3 buckets that are publicly accessible:

- name: buckets-public-access-block

description: Amazon S3 provides Block public access (bucket settings) and Block public access (account settings) to help you manage public access to Amazon S3 resources. By default, S3 buckets and objects are created with public access disabled. However, an IAM principle with sufficient S3 permissions can enable public access at the bucket and/or object level. While enabled, Block public access (bucket settings) prevents an individual bucket, and its contained objects, from becoming publicly accessible. Similarly, Block public access (account settings) prevents all buckets, and contained objects, from becoming publicly accessible across the entire account.

resource: s3

filters:

- or:

- type: check-public-block

BlockPublicAcls: false

- type: check-public-block

BlockPublicPolicy: false

- type: check-public-block

IgnorePublicAcls: false

- type: check-public-block

RestrictPublicBuckets: false

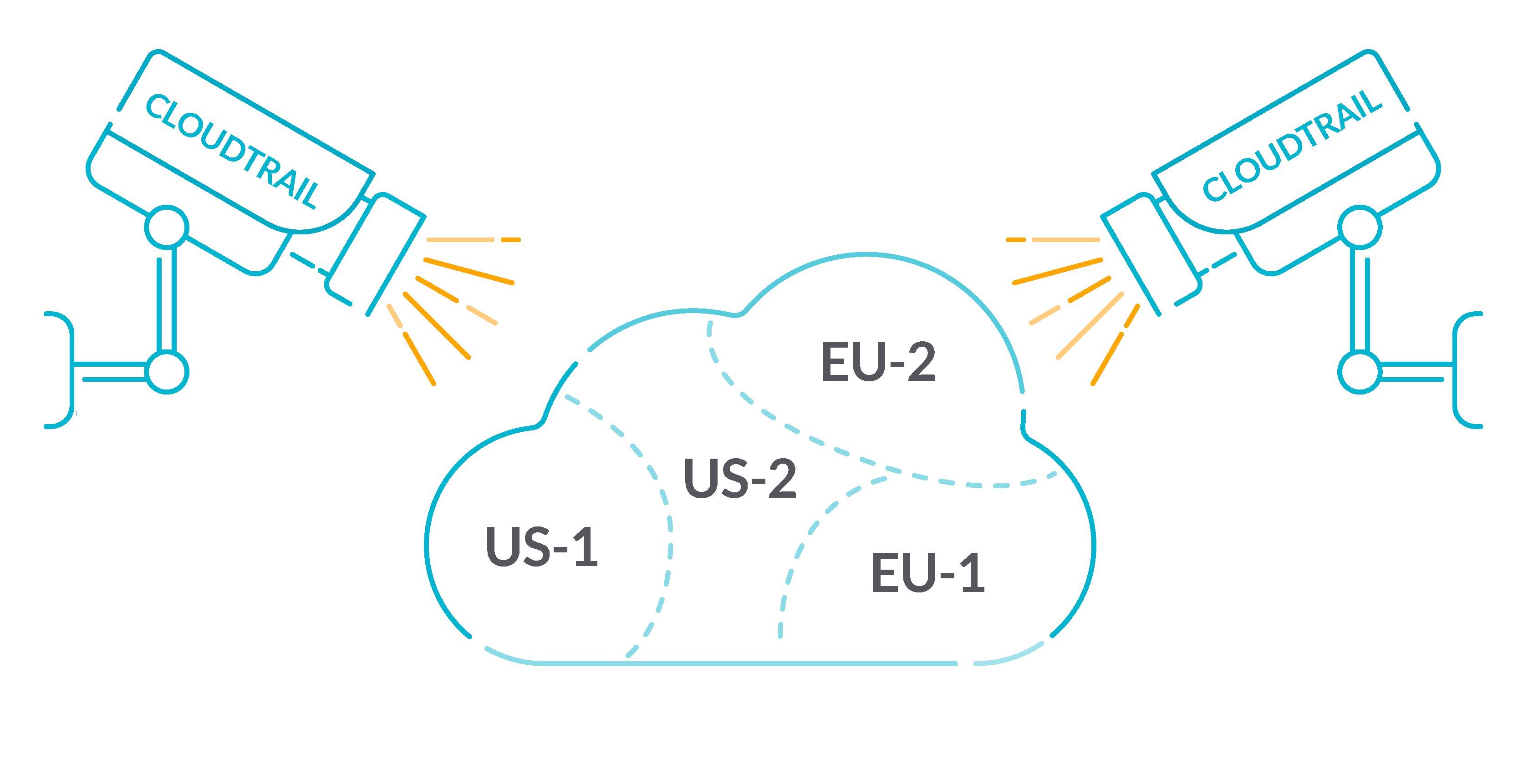
## AWS CloudTrail

AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.

The following section will help you configure CloudTrail to monitor your infrastructure across all your regions.

### 12.- Enable and configure CloudTrail with at least one multi-Region trail 🟥🟥🟥

CloudTrail provides a history of AWS API calls for an account, including API calls made from the AWS Management Console, AWS SDKs, and command line tools. The history also includes API calls from higher-level AWS services, such as AWS CloudFormation.

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-cloudtrail.png)

The **AWS API call history produced by CloudTrail enables security analysis**, **resource change tracking**, and **compliance auditing**. Multi-Region trails also provide the following benefits.

* A multi-Region trail helps to detect unexpected activity occurring in otherwise unused Regions.
* A multi-Region trail ensures that global service event logging is enabled for a trail by default. Global service event logging records events generated by AWS global services.
* For a multi-Region trail, management events for all read and write operations ensure that CloudTrail records management operations on all of an AWS account’s resources.

By default, CloudTrail trails that are created using the AWS Management Console are multi-Region trails.

List all trails available in the selected AWS region:

aws cloudtrail describe-trails

The output exposes each AWS CloudTrail trail along with its configuration details. If IsMultiRegionTrail config parameter value is false, the selected trail is not currently enabled for all AWS regions:

{

"trailList": [

{

"IncludeGlobalServiceEvents": true,

"Name": "ExampleTrail",

"TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/ExampleTrail",

"LogFileValidationEnabled": false,

"IsMultiRegionTrail": false,

"S3BucketName": "ExampleLogging",

"HomeRegion": "us-east-1"

}

]

}

Verify that all of your trails and make sure at least one is multi-Region.

### 13.- Enable encryption at rest with CloudTrail 🟨🟨

Check whether CloudTrail is configured to use the server-side encryption (SSE) AWS Key Management Service customer master key (CMK) encryption.

The check passes if the KmsKeyId is defined. For an added layer of security for your sensitive CloudTrail log files, you should use server-side encryption with AWS KMS–managed keys (SSE-KMS) for your CloudTrail log files for encryption at rest. Note that by default, the log files delivered by CloudTrail to your buckets are encrypted by Amazon server-side encryption with Amazon S3-managed encryption keys (SSE-S3).

You can check that the logs are encrypted with the following Cloud Custodian rule:

- name: cloudtrail-logs-encrypted-at-rest

description: AWS CloudTrail is a web service that records AWS API calls for an account and makes those logs available to users and resources in accordance with IAM policies. AWS Key Management Service (KMS) is a managed service that helps create and control the encryption keys used to encrypt account data, and uses Hardware Security Modules (HSMs) to protect the security of encryption keys. CloudTrail logs can be configured to leverage server side encryption (SSE) and KMS customer created master keys (CMK) to further protect CloudTrail logs. It is recommended that CloudTrail be configured to use SSE-KMS.

resource: cloudtrail

filters:

- type: value

key: KmsKeyId

value: absent

You can remediate it using the AWS Console like this:

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/cloudtrail/>.
2. In the left navigation panel, select **Trails**.
3. Under the **Name** column, select the trail name that you need to update.
4. Click the pencil icon next to the S3 section to edit the trail bucket configuration.
5. Under **S3 bucket\*** click **Advanced**.
6. Select **Yes** next to **Encrypt log files** to encrypt your log files with SSE-KMS using a Customer Master Key (CMK).
7. Select **Yes** next to **Create a new KMS key** to create a new CMK and enter a name for it, or otherwise select **No** to use an existing CMK encryption key available in the region.
8. Click **Save** to enable SSE-KMS encryption.

### 14.- Enable CloudTrail log file validation 🟨🟨

CloudTrail log file validation creates a digitally signed digest file that contains a hash of each log that CloudTrail writes to Amazon S3. You can use these digest files to determine whether a log file was changed, deleted, or unchanged after CloudTrail delivered the log.

It is recommended that you enable file validation on all trails. Log file validation provides additional integrity checks of CloudTrail logs.

To check this in the AWS Console proceed as follows:

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/cloudtrail/>.
2. In the left navigation panel, select **Trails**.
3. Under the **Name** column, select the trail name that you need to examine.
4. Under **S3** section, check for **Enable log file validation** status:
5. Enable log file validation status. If the feature status is set to **No**, then the selected trail does not have log file integrity validation enabled. If this is the case, fix it:
   1. Click the pencil icon next to the **S3** section to edit the trail bucket configuration.
   2. Under **S3 bucket\*** click **Advanced** and search for the **Enable log file validation** configuration status.
   3. Select **Yes** to enable log file validation, and then click **Save**.

Learn more about [security best practices in AWS Cloudtrail](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/best-practices-security.html).

## AWS Config

AWS Config provides a detailed view of the resources associated with your AWS account, including how they are configured, how they are related to one another, and how the configurations and their relationships have changed over time.

### 15.- Verify AWS Config is enabled 🟥🟥🟥

The AWS Config service performs configuration management of supported AWS resources in your account and delivers log files to you. The recorded information includes the configuration item (AWS resource), relationships between configuration items, and any configuration changes between resources.

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-config.png)

It is recommended that you enable AWS Config in all Regions. The AWS configuration item history that AWS Config captures enables security analysis, resource change tracking, and compliance auditing.

Get the status of all configuration recorders and delivery channels created by the Config service in the selected region:

aws configservice --region REGION get-status

The output from the previous command shows the status of all AWS Config delivery channels and configuration recorders available. If AWS Config is not enabled, the list for both configuration recorders and delivery channels are shown empty:

Configuration Recorders:

Delivery Channels:

Or, if the service was previously enabled but is now disabled, the status should be set to OFF:

Configuration Recorders:

name: default

recorder: OFF

Delivery Channels:

name: default

last stream delivery status: NOT\_APPLICABLE

last history delivery status: SUCCESS

last snapshot delivery status: SUCCESS

To remediate this, after you enable AWS Config, configure it to record all resources.

1. Open the AWS Config console at <https://console.aws.amazon.com/config/>.
2. Select the Region to configure AWS Config in.
3. If you haven’t used AWS Config before, see [Getting Started](https://docs.aws.amazon.com/config/latest/developerguide/getting-started.html) in the AWS Config Developer Guide.
4. Navigate to the Settings page from the menu, and do the following:
   1. Choose Edit.
   2. Under Resource types to record, select Record all resources supported in this region and Include global resources (e.g., AWS IAM resources).
   3. Under Data retention period, choose the default retention period for AWS Config data, or specify a custom retention period.
   4. Under AWS Config role, either choose Create AWS Config service-linked role or choose Choose a role from your account and then select the role to use.
   5. Under Amazon S3 bucket, specify the bucket to use or create a bucket and optionally include a prefix.
   6. Under Amazon SNS topic, select an Amazon SNS topic from your account or create one. For more information about Amazon SNS, see the [Amazon Simple Notification Service Getting Started Guide](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html).
5. Choose Save.

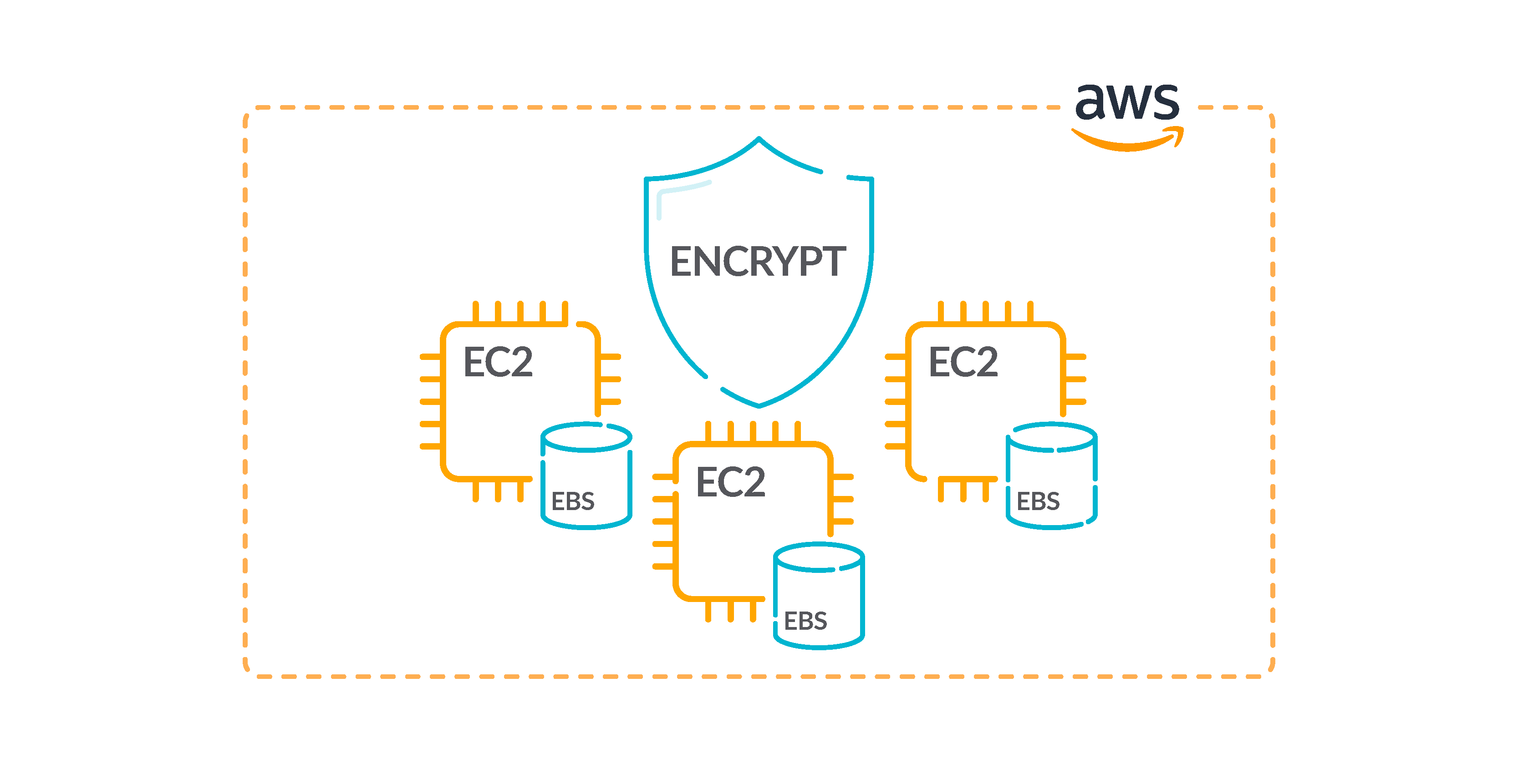
To go deeper, follow the [AWS security best practices for AWS Config](https://docs.aws.amazon.com/config/latest/developerguide/security-best-practices.html).

## Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems. Therefore, EC2 is one of the core services of AWS and it is necessary to know the best security practices and [how to secure EC2](https://sysdig.com/blog/aws-ec2-security-cloudtrail-sysdig/).

### 16.- Ensure attached EBS volumes are encrypted at rest 🟥🟥🟥

It is to check whether the EBS volumes that are in an attached state are encrypted. To pass this check, **EBS volumes must be in use and encrypted**. If the EBS volume is not attached, then it is not subject to this check.

[](https://sysdig.com/wp-content/uploads/aws-security-best-practices-ec2.png)

For an added layer of security to your sensitive data in EBS volumes, you should enable EBS encryption at rest. Amazon EBS encryption offers a straightforward encryption solution for your EBS resources that doesn’t require you to build, maintain, and secure your own key management infrastructure. It uses KMS keys when creating encrypted volumes and snapshots.

Run the describe-volumes command to determine if your EC2 Elastic Block Store volume is encrypted:

aws ec2 describe-volumes

--filters Name=attachment.instance-id, Values=INSTANCE\_ID

The command output should reveal the instance EBS volume encryption status (true for enabled, false for disabled).

There is no direct way to encrypt an existing unencrypted volume or snapshot. You can only encrypt a new volume or snapshot when you create it.

If you enable encryption by default, Amazon EBS encrypts the resulting new volume or snapshot by using your default key for Amazon EBS encryption. Even if you have not enabled encryption by default, you can enable encryption when you create an individual volume or snapshot. In both cases, you can override the default key for Amazon EBS encryption and choose a symmetric customer managed key.

### 17.- Enable VPC flow logging in all VPCs 🟩

With the VPC Flow Logs feature, you can capture information about the IP address traffic going to and from network interfaces in your VPC. After you create a flow log, you can view and retrieve its data in CloudWatch Logs. To reduce cost, you can also send your flow logs to Amazon S3.

It is recommended that you enable flow logging for packet rejects for VPCs. Flow logs provide visibility into network traffic that traverses the VPC and can detect anomalous traffic or provide insight during security workflows. By default, the record includes values for the different components of the IP address flow, including the source, destination, and protocol.

- name: flow-logs-enabled

description: 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. After you've created a flow log, you can view and retrieve its data in Amazon CloudWatch Logs. It is recommended that VPC Flow Logs be enabled for packet 'Rejects' for VPCs.

resource: vpc

filters:

- not:

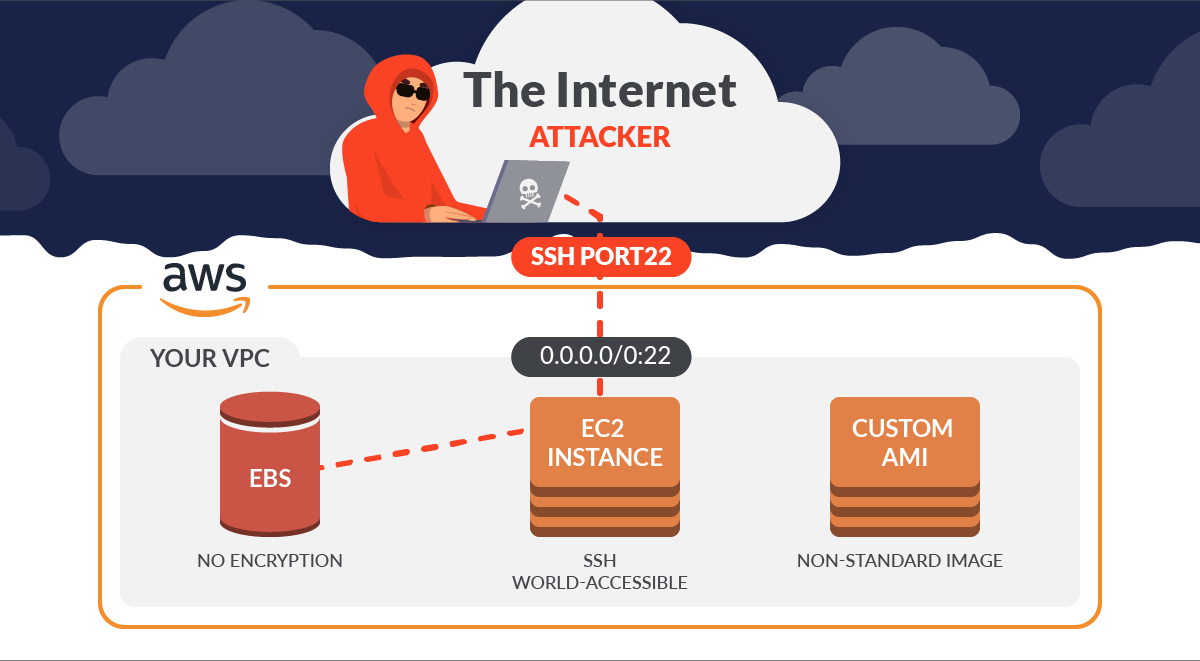
- type: flow-logs

enabled: true

### 18.- Confirm the VPC default security group does not allow inbound and outbound traffic 🟩

The rules for the default security group allow all outbound and inbound traffic from network interfaces (and their associated instances) that are assigned to the same security group.

**We do not recommend using the default security group**. Because the default security group cannot be deleted, you should change the default security group rules setting to restrict inbound and outbound traffic. This prevents unintended traffic if the default security group is accidentally configured for resources, such as EC2 instances.

[](https://sysdig.com/wp-content/uploads/AWS-security-best-practices-vpc.png)

Get the description of the default security group within the selected region:

aws ec2 describe-security-groups

--region REGION

--filters Name=group-name,Values='default'

--output table

--query 'SecurityGroups[\*].IpPermissions[\*].IpRanges'

If this command does not return any output, then the default security group does not allow public inbound traffic. Otherwise, it should return the inbound traffic source IPs defined, as in the following example:

------------------------

|DescribeSecurityGroups|

+----------------------+

| CidrIp |

+----------------------+

| 0.0.0.0/0 |

| ::/0 |

| 1.2.3.4/32 |

| 1.2.3.5/32 |

+----------------------+

If the IPs returned are 0.0.0.0/0 or ::/0, then the selected default security group is allowing public inbound traffic. We’ve explained previously [what the real threats are when securing SSH on EC2](https://sysdig.com/blog/aws-secure-ssh-ec2-threats/).

To remediate this issue,**create new security groups and assign those security groups to your resources**. To prevent the default security groups from being used, remove their inbound and outbound rules.

### 19.- Enable EBS default encryption 🟥🟥🟥

When encryption is enabled for your account, Amazon EBS volumes and snapshot copies are encrypted at rest. This adds an additional layer of protection for your data. For more information, see Encryption by default in the Amazon [EC2 User Guide](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/data-protection.html) for Linux Instances.

Note that following instance types do not support encryption: R1, C1, and M1.

Run the get-ebs-encryption-by-default command to know whether EBS encryption by default is enabled for your AWS cloud account in the selected region:

aws ec2 get-ebs-encryption-by-default

--region REGION

--query 'EbsEncryptionByDefault'

If the command returns false, the encryption of data at rest by default for new EBS volumes is not enabled in the selected AWS region. Fix it with the following command:

aws ec2 enable-ebs-encryption-by-default

--region REGION

## AWS Database Migration Service (DMS)

AWS Database Migration Service (AWS DMS) is a cloud service that makes it easy to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores. You can use AWS DMS to migrate your data into the AWS Cloud or between combinations of cloud and on-premises setups.

### 20.- Verify AWS Database Migration Service replication instances are not public 🟥🟥🟥

Ensure that your Amazon Database Migration Service (DMS) is not publicly accessible from the Internet in order to avoid exposing private data and minimize security risks. **A DMS replication instance should have a private IP address and the Publicly Accessible feature disabled** when both the source and the target databases are in the same network that is connected to the instance’s VPC through a VPN, VPC peering connection, or using an AWS Direct Connect dedicated connection.

1. Sign in to AWS Management Console at <https://console.aws.amazon.com/dms/>.
2. In the left navigation panel, choose **Replication instances**.
3. Select the DMS replication instance that you want to examine to open the panel with the resource configuration details.
4. Select the **Overview** tab from the dashboard bottom panel and check the **Publicly accessible** configuration attribute value. If the attribute value is set to **Yes**, the selected Amazon DMS replication instance is accessible outside the Virtual Private Cloud (VPC) and can be exposed to security risks. To fix it, do the following:
   1. Click the **Create replication instance** button from the dashboard top menu to initiate the launch process.
   2. On **Create replication instance** page, perform the following:
      1. Uncheck **Publicly accessible** checkbox to disable the public access to the new replication instance. If this setting is disabled, Amazon DMS will not assign a public IP address to the instance at creation and you will not be able to connect to the source/target databases outside the VPC.
      2. Provide a unique name for the new replication instance within the **Name** box, then configure the rest of the instance settings using the configuration information copied at step No. 5.
      3. Click **Create replication instance** to launch your new Amazon DMS instance.
5. Update your database migration plan by developing a new migration task to include the newly created AWS DMS replication instance.
6. To stop adding charges for the old replication instance:
   1. Select the old DMS instance, then click the **Delete** button from the dashboard top menu.
   2. Within the **Delete replication instance** dialog box, review the instance details then click **Delete** to terminate the selected DMS resource.

 Repeat step Nos. 3 and 4 for each AWS DMS replication instance provisioned in the selected region.

 Change the region from the console navigation bar and repeat the process for all the other regions.

Learn more about [AWS security best practices for AWS Database Migration Service](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_BestPractices.html).

## Amazon Elastic Block Store (EBS)

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes, or use them in any way you would use a block device (such as a hard drive).

You can dynamically change the configuration of a volume attached to an instance.

### 21.- Ensure Amazon EBS snapshots are not public, or to be restored by anyone 🟥🟥🟥

EBS snapshots are used to back up the data on your EBS volumes to Amazon S3 at a specific point in time. You can use the snapshots to restore previous states of EBS volumes. It is rarely acceptable to share a snapshot with the public. **Typically, the decision to share a snapshot publicly was made in error or without a complete understanding of the implications**. This check helps ensure that all such sharing was fully planned and intentional.

Get the list of all EBS volume snapshots:

aws ec2 describe-snapshots

--region REGION

--owner-ids ACCOUNT\_ID

--filters Name=status,Values=completed

--output table

--query 'Snapshots[\*].SnapshotId'

For each snapshot, check its createVolumePermission attribute:

aws ec2 describe-snapshot-attribute

--region REGION

--snapshot-id SNAPSHOT\_ID

--attribute createVolumePermission

--query 'CreateVolumePermissions[]'

The output from the previous command returns information about the permissions for creating EBS volumes from the selected snapshot:

{

"Group": "all"

}

If the command output is "Group": "all", the snapshot is accessible to all AWS accounts and users. If this is the case, take your time to run this command to fix it:

aws ec2 modify-snapshot-attribute

--region REGION

--snapshot-id SNAPSHOT\_ID

--attribute createVolumePermission

--operation-type remove

--group-names all

## Amazon OpenSearch Service

Amazon OpenSearch Service is a managed service that makes it easy to deploy, operate, and scale OpenSearch clusters in the AWS Cloud. Amazon OpenSearch Service is the successor to Amazon Elasticsearch Service and supports OpenSearch and legacy Elasticsearch OSS (up to 7.10, the final open source version of the software). When you create a cluster, you have the option of which search engine to use.

### 22.- Ensure Elasticsearch domains have encryption at rest enabled 🟥🟥🟥

For an added layer of security for your sensitive data in OpenSearch, you should configure your OpenSearch to be encrypted at rest. Elasticsearch domains offer encryption of data at rest. The feature uses AWS KMS to store and manage your encryption keys. To perform the encryption, it uses the Advanced Encryption Standard algorithm with 256-bit keys (AES-256).

List all Amazon OpenSearch domains currently available:

aws es list-domain-names --region REGION

Now determine if data-at-rest encryption feature is enabled with:

aws es describe-elasticsearch-domain

--region REGION

--domain-name DOMAIN\_NAME

--query 'DomainStatus.EncryptionAtRestOptions'

If the Enabled flag is false, the data-at-rest encryption is not enabled for the selected Amazon ElasticSearch domain. Fix it with:

aws es create-elasticsearch-domain

--region REGION

--domain-name DOMAIN\_NAME

--elasticsearch-version 5.5

--elasticsearch-cluster-config InstanceType=m4.large.elasticsearch,InstanceCount=2

--ebs-options EBSEnabled=true,VolumeType=standard,VolumeSize=200

--access-policies file://source-domain-access-policy.json

--vpc-options SubnetIds=SUBNET\_ID,SecurityGroupIds=SECURITY\_GROUP\_ID

--encryption-at-rest-options Enabled=true,KmsKeyId=KMS\_KEY\_ID

Once the new cluster is provisioned, upload the existing data (exported from the original cluster) to the newly created cluster.

After all the data is uploaded, it is safe to remove the unencrypted OpenSearch domain to stop incurring charges for the resource:

aws es delete-elasticsearch-domain

--region REGION

--domain-name DOMAIN\_NAME

## Amazon SageMaker

Amazon SageMaker is a fully-managed machine learning service. With Amazon SageMaker, data scientists and developers can quickly build and train machine learning models, and then deploy them into a production-ready hosted environment.

### 23.- Verify SageMaker notebook instances do not have direct internet access 🟨🟨

If you configure your SageMaker instance without a VPC, then, by default, direct internet access is enabled on your instance. You should configure your instance with a VPC and change the default setting to Disable — Access the internet through a VPC.

To train or host models from a notebook, you need internet access. To enable internet access, **make sure that your VPC has a NAT gateway and your security group allows outbound connections**. To learn more about how to connect a notebook instance to resources in a VPC, see “[Connect a notebook instance to resources in a VPC](https://docs.amazonaws.cn/en_us/sagemaker/latest/dg/notebook-interface-endpoint.html)” in the Amazon SageMaker Developer Guide.

You should also ensure that access to your SageMaker configuration is limited to only authorized users. Restrict users’ IAM permissions to modify SageMaker settings and resources.

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/sagemaker/>.
2. In the navigation panel, under **Notebook**, choose **Notebook instances**.
3. Select the SageMaker notebook instance that you want to examine and click on the instance name (link).
4. On the selected instance configuration page, within the Network section, check for any VPC subnet IDs and security group IDs. If these network configuration details are not available, instead the following status is displayed: “**No custom VPC settings applied**.” The notebook instance is not running inside a VPC network, therefore you can follow the steps described in this conformity rule to deploy the instance within a VPC. Otherwise, if the notebook instance is running inside a VPC, check the **Direct internet access** configuration attribute value. If the attribute value is set to **Enabled**, the selected Amazon SageMaker notebook instance is publicly accessible.

If the notebook has direct internet access enabled, fix it by recreating it with this CLI command:

aws sagemaker create-notebook-instance

--region REGION

--notebook-instance-name NOTEBOOK\_INSTANCE\_NAME

--instance-type INSTANCE\_TYPE

--role-arn ROLE\_ARN

--kms-key-id KMS\_KEY\_ID

--subnet-id SUBNET\_ID

--security-group-ids SECURITY\_GROUP\_ID

--direct-internet-access Disabled

## AWS Lambda

With AWS Lambda, you can run code without provisioning or managing servers. You pay only for the compute time that you consume — there’s no charge when your code isn’t running. You can run code for virtually any type of application or backend service — all with zero administration.

Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

It is important to mention the problems that could occur if we do not secure or audit the code we execute in our lambda functions, as you could be the [initial access for attackers](https://sysdig.com/blog/exploit-mitigate-aws-lambdas-mitre/).

### 24.- Use supported runtimes for Lambda functions 🟨🟨

This AWS security best practice recommends checking that the Lambda function settings for runtimes match the expected values set for the supported runtimes for each language. This control checks function settings for the following runtimes: nodejs16.x, nodejs14.x, nodejs12.x, python3.9, python3.8, python3.7, ruby2.7, java11, java8, java8.al2, go1.x, dotnetcore3.1, and dotnet6.

The AWS Config rule ignores functions that have a package type of image.

Lambda runtimes are built around a combination of operating system, programming language, and software libraries that are subject to maintenance and security updates. When a runtime component is no longer supported for security updates, Lambda deprecates the runtime. Even though you cannot create functions that use the deprecated runtime, the function is still available to process invocation events. Make sure that your Lambda functions are current and do not use out-of-date runtime environments.

Get the names of all Amazon Lambda functions available in the selected AWS cloud region:

aws lambda list-functions

--region REGION

--output table

--query 'Functions[\*].FunctionName'

Now examine the runtime information available for each functions:

aws lambda get-function-configuration

--region REGION

--function-name FUNCTION\_NAME

--query 'Runtime'

Compare the value returned with the updated list of Amazon Lambda runtimes supported by AWS, as well as the end of support plan listed in the AWS documentation.

If the runtime is unsupported, fix it to use the latest runtime version. For example:

aws lambda update-function-configuration

--region REGION

--function-name FUNCTION\_NAME

--runtime "nodejs16.x"

## AWS Key Management Service (AWS KMS)

AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud. AWS KMS keys and functionality are used by other AWS services, and you can use them to protect data in your own applications that use AWS.

### 25.- Do not unintentionally delete AWS KMS keys 🟨🟨

KMS keys cannot be recovered once deleted. Data encrypted under a KMS key is also permanently unrecoverable if the KMS key is deleted. If meaningful data has been encrypted under a KMS key scheduled for deletion, consider decrypting the data or re-encrypting the data under a new KMS key unless you are intentionally performing a cryptographic erasure.

When a KMS key is scheduled for deletion, a mandatory waiting period is enforced to allow time to reverse the deletion if it was scheduled in error. The default waiting period is 30 days, but it can be reduced to as short as seven days when the KMS key is scheduled for deletion. During the waiting period, the scheduled deletion can be canceled and the KMS key will not be deleted.

List all Customer Master keys available in the selected AWS region:

aws kms list-keys --region REGION

Run the describe-key command for each CMK to identify any keys scheduled for deletion:

aws kms describe-key --key-id KEY\_ID

The output for this command shows the selected key metadata. If the KeyState value is set to PendingDeletion, the key is scheduled for deletion. But if this is not what you actually want (the most common case), unschedule the deletion with:

aws kms cancel-key-deletion --key-id KEY\_ID

## Amazon GuardDuty

Amazon GuardDuty is a continuous security monitoring service. Amazon GuardDuty can help to [identify unexpected and potentially unauthorized or malicious activity](https://sysdig.com/blog/hunting-malware-with-amazon-guardduty-and-sysdig/) in your AWS environment.

### 26.- Enable GuardDuty 🟨🟨

It is highly recommended that you enable GuardDuty in all supported AWS Regions. Doing so allows GuardDuty to generate findings about unauthorized or unusual activity, even in Regions that you do not actively use. This also allows GuardDuty to monitor CloudTrail events for global AWS services, such as IAM.

List the IDs of all the existing Amazon GuardDuty detectors. A detector is an object that represents the AWS GuardDuty service. A detector must be created in order for GuardDuty to become operational:

aws guardduty list-detectors

--region REGION

--query 'DetectorIds'

If the list-detectors command output returns an empty array, then there are no GuardDuty detectors available. In this instance, the Amazon GuardDuty service is not enabled within your AWS account. If this is the case, create a detector with the following command:

aws guardduty create-detector

--region REGION

--enable

Once the detector is enabled, it will start to pull and analyze independent streams of data from AWS CloudTrail, VPC flow logs, and [DNS logs in order to generate findings](https://sysdig.com/blog/dns-security-cloud-protection/).

## AWS compliance standards & benchmarks

**Setting up and maintaining your AWS infrastructure to keep it secure is a never-ending effort** that will require a lot of time.

For this, you will be better off following the compliance standard(s) relevant to your industry, since they provide all the requirements needed to effectively secure your cloud environment.

Because of the ongoing nature of securing your environment and complying with a security standard, you might also want to recurrently run policies, such as **CIS Amazon Web Services Foundations Benchmark**, which will audit your system and report any non-conformity it finds.