SWINBURNE UNIVERSITY OF TECHNOLOGY

COS80001

Cloud Computing Architecture

Lecture 06 Security

includes material from

ACF Module Security

ACA Module 10 – Well-architected Security

Reminders

- ☐Assignment 1B due by week 7
- ☐Assignment 2 to be released

- ☐ First MCQ exam
- ☐ Access from Canvas
- ☐ You must pass MCQs to pass the unit
- ☐ Practice test available.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Last week

- Relational vs non-relational 'NoSQL' databases
  - High-level Comparison
  - NoSQL data models
- AWS Database Services
  - RDS
  - Dynamo DB
  - Redshift data warehouse
  - (Aurora)

Quizzes:
ACF 2.0.4 Database
ACA Mod 7 Web Scale media

SWINBURNE UNIVERSITY OF TECHNOLOGY

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - Users, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

Quizzes:
ACF Security
ACA Mod 10 Well Architected Security

SWINBURNE UNIVERSITY OF TECHNOLOGY

Extras: AWS Security and Auditing Services

- Optional Extras (for ACA Certification)
  - AWS Trusted Advisor
  - AWS Security and Compliance Programs
  - AWS Security Resources

SWINBURNE UNIVERSITY OF TECHNOLOGY

This week - Security

- **Access Control concepts**
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - Users, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

Access Control Concepts: **Authentication** & Authorisation

- **Authentication** – *Establish 'your' identity*.
  - Username and password
  - Access key ID and secret access key (programmatic)
  - Federated identity with Single Sign on (SSO)
    - Identity providers: e.g. LDAP, Microsoft, Okta, …
  - OpenID Connect: e.g. Google confirming your email address
  - SAML (Security Assertion Markup Language): e.g. enterprises
  - Multi-factor Authentication, One-time passwords, …
  - Web applications – SPAs (Single page apps) and multi-page sessions
    - Cookie-based
    - Token-based - JWT

SWINBURNE UNIVERSITY OF TECHNOLOGY

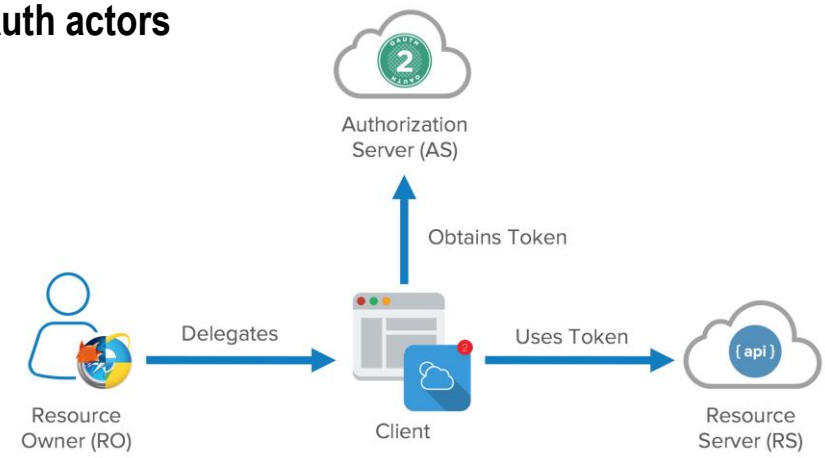
Access Control Concepts: Authentication & **Authorisation**

- **Authorisation** – *What resources are you/app allowed to access*?
- Control needed at many levels
  - Network control: e.g. Firewalls, ACLs, Security groups (Lecture 3)
  - File system permissions: e.g. Unix permissions -rwxr-xr-x
  - OS policies: e.g. Windows resources and user policies
- Authorisation inside and outside organizational boundaries
- Applications ↑ web based, ↑ decentralized, ↑ RESTful APIs
  → need to ***delegate authorisation***
  - OAuth 2.0 standard - delegates authorization to devices, apps etc using tokens

SWINBURNE UNIVERSITY OF TECHNOLOGY

Example: Web based Authorisation Delegation

- **OAuth actors**



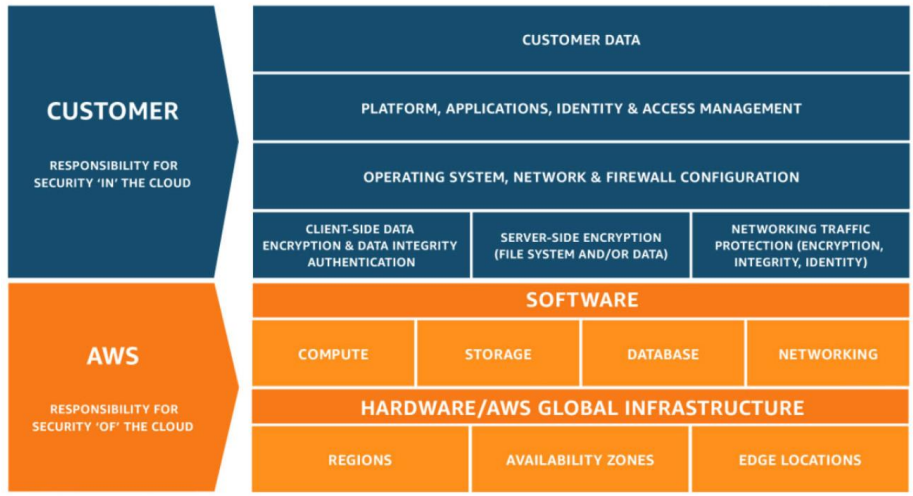
From: https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth

This week - Security

- Access Control concepts
- **Security in the Cloud: AWS Shared Responsibility Model**
- AWS IAM
  - Users, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

AWS shared responsibility model

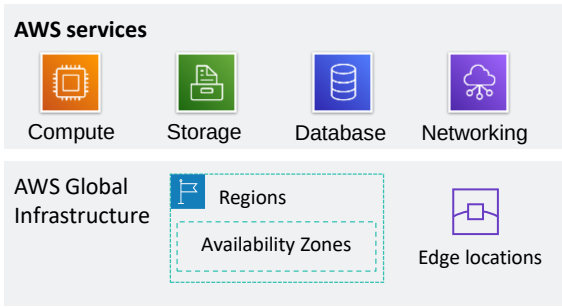


Security and compliance are a shared responsibility between AWS and the customer. This shared responsibility model is designed to help relieve the customer's operational burden. At the same time, to provide the flexibility and customer control that enables the deployment of customer solutions on AWS, the customer remains responsible for some aspects of the overall security. The differentiation of who is responsible for what is commonly referred to as *security "of" the cloud* versus *security "in" the cloud*.

AWS operates, manages, and controls the components from the software virtualization layer down to the physical security of the facilities where AWS services operate. **AWS is responsible** for protecting the infrastructure that runs all the services that are offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run the AWS Cloud services.

The customer is responsible for the encryption of data at rest and data in transit. The customer should also ensure that the network is configured for security and that security credentials and logins are managed safely. Additionally, the customer is responsible for the configuration of security groups and the configuration of the operating system that run on compute instances that they launch (including updates and security patches).

AWS responsibility: Security *of* the cloud



AWS responsibilities:

- Physical security of data centers
  - Controlled, need-based access
- Hardware and software infrastructure
  - Storage decommissioning, host operating system (OS) access logging, and auditing
- Network infrastructure
  - Intrusion detection

- Virtualization infrastructure
  - Instance isolation

AWS is responsible for security ***of*** the cloud. But what does that mean?

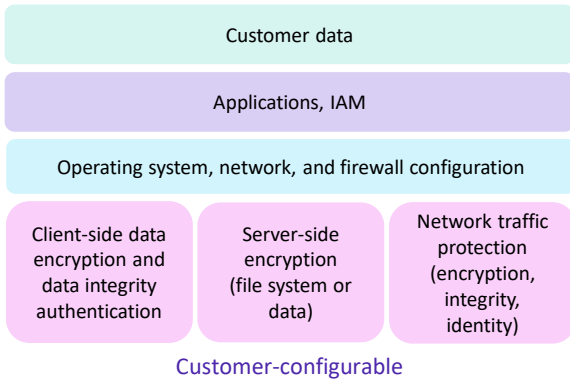
Under the AWS shared responsibility model, AWS operates, manages, and controls the components from the bare metal host operating system and hypervisor virtualization layer down to the physical security of the facilities where the services operate. It means that AWS is responsible for protecting the global infrastructure that runs all the services that are offered in the AWS Cloud. The global infrastructure includes AWS Regions, Availability Zones, and edge locations.

AWS is responsible for the physical infrastructure that hosts your resources, including:

- **Physical security of data centers** with controlled, need-based access; located in nondescript facilities, with 24/7 security guards; two-factor authentication; access logging and review; video surveillance; and disk degaussing and destruction.
- **Hardware infrastructure**, such as servers, storage devices, and other appliances that AWS relies on.
- **Software infrastructure**, which hosts operating systems, service applications, and virtualization software.
- **Network infrastructure**, such as routers, switches, load balancers, firewalls, and cabling. AWS also continuously monitors the network at external boundaries, secures access points, and provides redundant infrastructure with intrusion detection.

Protecting this infrastructure is the top priority for AWS. While you cannot visit AWS data centers or offices to see this protection firsthand, Amazon provides several reports from third-party auditors who have verified our compliance with a variety of computer security standards and regulations.

Customer responsibility: Security *in* the cloud



Customer responsibilities:

- Amazon Elastic Compute Cloud (Amazon EC2) instance operating system
  - Including patching, maintenance
- Applications
  - Passwords, role-based access, etc.
- Security group configuration
- OS or host-based firewalls
  - Including intrusion detection or prevention systems
- Network configurations
- Account management
  - Login and permission settings for each user

While the cloud infrastructure is secured and maintained by AWS, customers are responsible for security of everything they put ***in*** the cloud.

The **customer is responsible** for what is implemented by using AWS services and for the applications that are connected to AWS. The security steps that you must take depend on the services that you use and the complexity of your system.

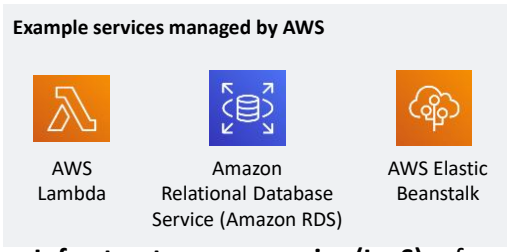
Customer responsibilities include selecting and securing any instance operating systems, securing the applications that are launched on AWS resources, security group configurations, firewall configurations, network configurations, and secure account management.

When customers use AWS services, they maintain complete control over their content. Customers are responsible for managing critical content security requirements, including:

- What content they choose to store on AWS
- Which AWS services are used with the content
- In what country that content is stored
- The format and structure of that content and whether it is masked, anonymized, or encrypted
- Who has access to that content and how those access rights are granted, managed, and revoked

Customers retain control of what security they choose to implement to protect their own data, environment, applications, IAM configurations, and operating systems.

Service characteristics and security responsibility



Infrastructure as a service (IaaS)

- Customer has more flexibility over configuring networking and storage settings
- Customer is responsible for managing more aspects of the security
- Customer configures the access controls

Platform as a service (PaaS)

- Customer does not need to manage the underlying infrastructure
- AWS handles the operating system, database patching, firewall configuration, and disaster recovery
- Customer can focus on managing code or data

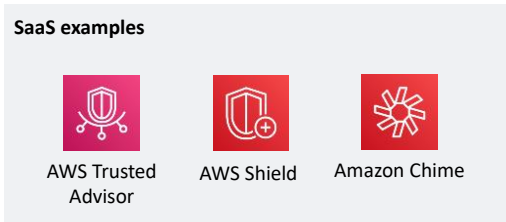
Infrastructure as a service (IaaS) refers to services that provide basic building blocks for cloud IT, typically including access to configure networking, computers (virtual or on dedicated hardware), and data storage space. Cloud services that can be characterized as IaaS **provide the customer with the highest level of flexibility and management control** over IT resources. IaaS services are most similar to existing on-premises computing resources that many IT departments are familiar with today.

AWS services—such as **Amazon EC2**—can be categorized as **IaaS** and thus **require the customer to perform all necessary security configuration and management tasks**. Customers who deploy EC2 instances are responsible for managing the guest operating system (including updates and security patches), any application software that is installed on the instances, and the configuration of the security groups that were provided by AWS.

Platform as a service (PaaS) refers to services that remove the need for the customer to manage the underlying infrastructure (hardware, operating systems, etc.). PaaS services enable the customer to focus entirely on deploying and managing applications. Customers don't need to worry about resource procurement, capacity planning, software maintenance, or patching.

AWS services such as **AWS Lambda** and **Amazon RDS** can be categorized as **PaaS** because **AWS operates the infrastructure layer, the operating system, and platforms**. Customers only need to access the endpoints to store and retrieve data. With PaaS services, customers are responsible for managing their data, classifying their assets, and applying the appropriate permissions. However, these service act more like managed services, with AWS handling a larger portion of the security requirements. For these services, AWS handles basic security tasks—such as operating system and database patching, firewall configuration, and disaster recovery.

Service characteristics and security responsibility (continued)



Software as a service (SaaS)

- Software is centrally hosted
- Licensed on a subscription model or pay-as-you-go basis.
- Services are typically accessed via web browser, mobile app, or application programming interface (API)
- Customers do not need to manage the infrastructure that supports the service

Software as a service (SaaS) refers to services that provide centrally hosted software that is typically accessible via a web browser, mobile app, or application programming interface (API). The licensing model for SaaS offerings is typically subscription or pay as you go. With SaaS offerings, customers do not need to manage the infrastructure that supports the service. Some AWS services—such as **AWS Trusted Advisor, AWS Shield,** and **Amazon Chime**—could be categorized as SaaS offerings, given their characteristics.

AWS Trusted Advisor is an online tool that analyzes your AWS environment and provides real-time guidance and recommendations to help you provision your resources by following AWS best practices. The Trusted Advisor service is offered as part of your AWS Support plan. Some of the Trusted Advisor features are free to all accounts, but Business Support and Enterprise Support customers have access to the full set of Trusted Advisor checks and recommendations.

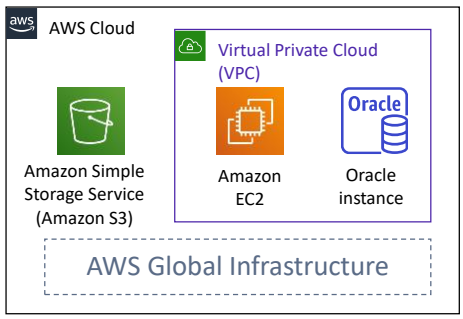
AWS Shield is a managed distributed denial of service (DDoS) protection service that safeguards applications running on AWS. It provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection. AWS Shield Advanced is available to all customers. However, to contact the DDoS Response Team, customers must have either Enterprise Support or Business Support from AWS Support.

Amazon Chime is a communications service that enables you to meet, chat, and place business calls inside and outside your organization, all using a single application. It is a pay-as-you-go communications service with no upfront fees, commitments, or long-term contracts.

Activity: Scenario 1 of 2

aws academy

Consider this deployment. Who is responsible – AWS or the customer?



1. Upgrades and patches to the operating system on the EC2 instance?
   - **ANSWER:** The customer
2. Physical security of the data center?
   - **ANSWER:** AWS
3. Virtualization infrastructure?
   - **ANSWER:** AWS
4. EC2 security group settings?
   - **ANSWER:** The customer
5. Configuration of applications that run on the EC2 instance?
   - **ANSWER:** The customer
6. Oracle upgrades or patches If the Oracle instance runs as an Amazon RDS instance?
   - **ANSWER:** AWS
7. Oracle upgrades or patches If Oracle runs on an EC2 instance?
   - **ANSWER:** The customer
8. S3 bucket access configuration?
   - **ANSWER:** The customer

Consider the case where a customer uses the AWS services and resources that are shown here. Who is responsible for maintaining security? AWS or the customer?

The customer uses Amazon Simple Storage Service (Amazon S3) to store data. The customer configured a virtual private cloud (VPC) with Amazon Virtual Private Cloud (Amazon VPC). The EC2 instance and the Oracle database instance that they created both run in the VPC.

In this example, the customer must manage the guest operating system (OS) that runs on the **EC2 instance**. Over time, the guest OS will need to be upgraded and have security patches applied. Additionally, any application software or utilities that the customer installed on the Amazon EC2 instance must also be maintained. The customer is responsible for configuring the AWS firewall (or security group) that is applied to the Amazon EC2 instance. The customer is also responsible for the **VPC** configurations that specify the network conditions in which the Amazon EC2 instance runs. These tasks are the same security tasks that IT staff would perform, no matter where their servers are located.

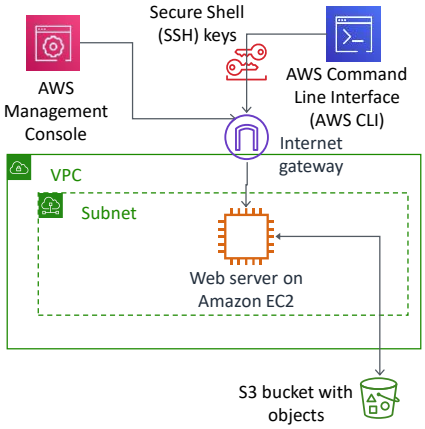
The Oracle instance in this example provides an interesting case study in terms of AWS or customer responsibility. **If the database runs on an EC2 instance**, then it is the customer's responsibility to apply Oracle software upgrades and patches. However, **if the database runs**

as an Amazon RDS instance, then it is the responsibility of AWS to apply Oracle software upgrades and patches. Because Amazon RDS is a managed database offering, time-consuming database administration tasks—which include provisioning, backups, software patching, monitoring, and hardware scaling—are handled by AWS. To learn more, see Best Practices for Running Oracle Database on AWS for details.

Activity: Scenario 2 of 2

aws academy

Consider this deployment. Who is responsible – AWS or the customer?



1. Ensuring that the AWS Management Console is not hacked?
   - **ANSWER:** AWS
2. Configuring the subnet?
   - **ANSWER:** The customer
3. Configuring the VPC?
   - **ANSWER:** The customer
4. Protecting against network outages in AWS Regions?
   - **ANSWER:** AWS
5. Securing the SSH keys
   - **ANSWER:** The customer
6. Ensuring network isolation between AWS customers' data?
   - **ANSWER:** AWS
7. Ensuring low-latency network connection between the web server and the S3 bucket?
   - **ANSWER:** AWS
8. Enforcing multi-factor authentication for all user logins?
   - **ANSWER:** The customer

Now, consider this additional case where a customer uses the AWS services and resources that are shown here. Who is responsible for maintaining security? AWS or the customer?

A customer uses Amazon S3 to store data. The customer configured a virtual private cloud (VPC) with Amazon VPC, and is running a web server on an EC2 instance in the VPC. The customer configured an internet gateway as part of the VPC so that the web server can be reached by using the AWS Management Console or the AWS Command Line Interface (AWS CLI). When the customer uses the AWS CLI, the connection requires the use of Secure Shell (SSH) keys.

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- **AWS IAM**
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

AWS Identity and Access Management (IAM)

- Use **IAM** to manage access to **AWS resources** –
  - A resource is an entity in an AWS account that you can work with
  - Example resources; An Amazon EC2 instance or an Amazon S3 bucket
- *Example* – Control who can terminate Amazon EC2 instances
- Define fine-grained access rights –
  - **Who** can access the resource
  - **Which** resources can be accessed and what can the user do to the resource
  - **How** resources can be accessed
- IAM is a no-cost AWS account feature

AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) allows you to control access to compute, storage, database, and application services in the AWS Cloud. IAM can be used to handle authentication, and to specify and enforce authorization policies so that you can specify which users can access which services.

IAM is a tool that centrally manages access to launching, configuring, managing, and terminating resources in your AWS account. It provides granular control over access to resources, including the ability to specify exactly which **API** calls the user is authorized to make to each service. Whether you use the AWS Management Console, the AWS CLI, or the AWS software development kits (SDKs), every call to an AWS service is an API call.

With IAM, you can manage *which* resources can be accessed by *who*, and *how* these resources can be accessed. You can grant different permissions to different people for different resources. For example, you might allow some users full access to Amazon EC2, Amazon S3, Amazon DynamoDB, Amazon Redshift, and other AWS services. However, for other users, you might allow read-only access to only a few S3 buckets. Similarly, you might grant permission to other users to administer only specific EC2 instances. You could also allow a few users to access only the account billing information, but nothing else.

IAM is a feature of your AWS account, and it is offered at no additional charge.

IAM: Essential components

IAM user

A **person** *or* **application** that can authenticate with an AWS account.

IAM group

A **collection of IAM users** that are granted identical authorization.

IAM policy

The document that defines **which resources can be accessed** and the **level of access** to each resource.

IAM role

Useful mechanism to grant a set of permissions for making AWS service requests.

To understand how to use IAM to secure your AWS account, it is important to understand the role and function of each of the four IAM components.

An **IAM user** is a person or application that is defined in an AWS account, and that must make API calls to AWS products. Each user must have a unique name (with no spaces in the name) within the AWS account, and a set of security credentials that is not shared with other users. These credentials are different from the AWS account root user security credentials. Each user is defined in one and only one AWS account.

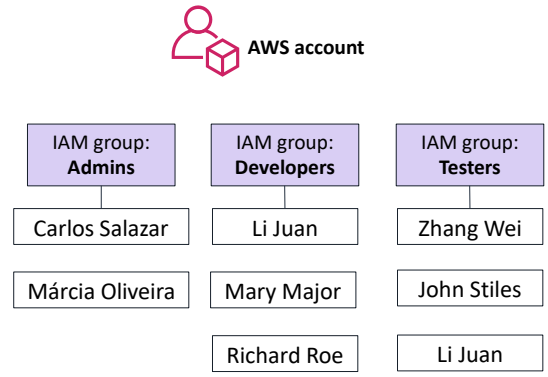
An **IAM group** is a collection of IAM users. You can use IAM groups to simplify specifying and managing permissions for multiple users.

An **IAM policy** is a document that defines permissions to determine what users can do in the AWS account. A policy typically grants access to specific resources and specifies what the user can do with those resources. Policies can also explicitly deny access.

An **IAM role** is a tool for granting temporary access to specific AWS resources in an AWS account.

IAM users and groups

- An **IAM group** is a collection of IAM users
- A group is used to grant the same permissions to multiple users
  - Permissions granted by attaching IAM *policy* or policies to the group
- A user can belong to multiple groups
- There is no default group
- Groups cannot be nested



An **IAM group** is a collection of IAM users. IAM groups offer a convenient way to specify permissions for a collection of users, which can make it easier to manage the permissions for those users.

For example, you could create an IAM group that is called *Developers* and attach an IAM policy or multiple IAM policies to the Developers group that grant the AWS resource access permissions that developers typically need. Any user that you then add to the Developer group will automatically have the permissions that are assigned to the group. In such a case, you do not need to attach the IAM policy or IAM policies directly to the user. If a new user joins your organization and should be granted developer privileges, you can simply add that user to the Developers group. Similarly, if a person changes jobs in your organization, instead of editing that user's permissions, simply remove the user from the group.

Important characteristics of IAM groups:

- A group can contain many users, and a user can belong to multiple groups.
- Groups cannot be nested. A group can contain only users, and a group cannot contain other groups.
- There is no default group that automatically includes all users in the AWS account. If you want to have a group with all account users in it, you need to create the group and add each new user to it.

Lab 1 Review: Tasks

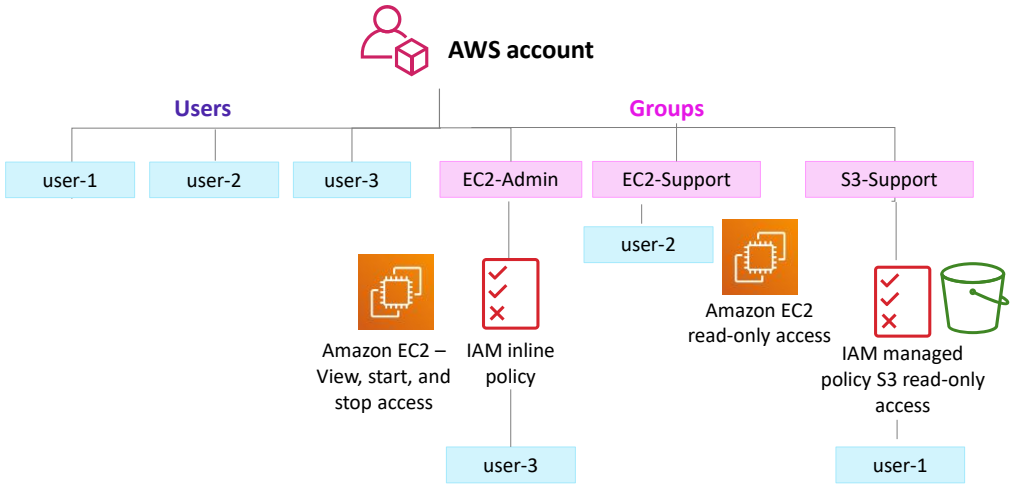
- Task 1: Explore the Users and Groups.
- Task 2: Add Users to Groups.
- Task 3: Sign-In and Test Users.

AWS Identity and Access Management (IAM)

In this hands-on lab, you will:

- Explore pre-created IAM users and groups.
- Inspect IAM policies as they are applied to the pre-created groups.
- Follow a real-world scenario and add users to groups that have specific capabilities enabled.
- Locate and use the IAM sign-in URL.
- Experiment with the effects of IAM policies on access to AWS resources.

Lab 1: Final product



The diagram shows the resources that your AWS account will have after you complete the lab steps. It also describes how the resources will be configured.

Authenticate as an IAM user to gain access

When you define an **IAM user**, you select what ***types of access*** the user is permitted to use.

Programmatic access

- Authenticate using:
  - Access key ID
  - Secret access key
- Provides AWS CLI and AWS SDK access

AWS CLI

AWS Tools and SDKs

AWS Management Console access

- Authenticate using:
  - 12-digit Account ID *or* alias
  - IAM user name
  - IAM password
- If enabled, **multi-factor authentication (MFA)** prompts for an authentication code.

AWS Management Console

Authentication is a basic computer security concept: a user or system must first prove their identity. Consider how you authenticate yourself when you go to the airport and you want to get through airport security so that you can catch your flight. In this situation, you must present some form of identification to the security official to prove who you are before you can enter a restricted area. A similar concept applies for gaining access to AWS resources in the cloud.

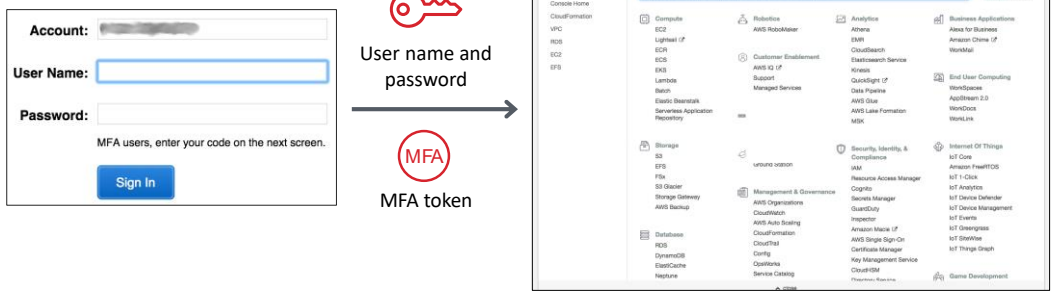
When you define an IAM user, you select what type of access the user is permitted to use to access AWS resources. You can assign two different types of access to users: programmatic access and AWS Management Console access. You can assign programmatic access only, console access only, or you can assign both types of access.

If you grant **programmatic access**, the IAM user will be required to present an **access key ID** and a **secret access key** when they make an AWS API call by using the AWS CLI, the AWS SDK, or some other development tool.

If you grant **AWS Management Console access**, the IAM user will be required to fill in the fields that appear in the browser login window. The user is prompted to provide either the 12-digit account ID or the corresponding account alias. The user must also enter their IAM user name and password. If **multi-factor authentication (MFA)** is enabled for the user, they will also be prompted for an authentication code.

IAM MFA

- MFA provides increased security.
- In addition to **user name** and **password**, MFA requires a unique authentication code to access AWS services.



AWS Management Console

AWS services and resources can be accessed by using the AWS Management Console, the AWS CLI, or through SDKs and APIs. For increased security, we recommend enabling MFA.

With MFA, users and systems must provide an **MFA token**—in addition to the regular sign-in credentials—before they can access AWS services and resources.

Options for generating the MFA authentication token include **virtual MFA-compliant applications** (such as Google Authenticator or Authy 2-Factor Authentication), **U2F security key devices**, and **hardware MFA devices**.

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- **AWS IAM**
  - User Authentication, Groups, Roles
  - **Authorization using Policies**
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

IAM roles

- An **IAM role** is an IAM identity with specific permissions
- Similar to an IAM user
  - Attach permissions policies to it
- Different from an IAM user
  - Not uniquely associated with one person
  - Intended to be *assumable* by a **person**, **application**, or **service**
- Role provides *temporary* security credentials
- Examples of how IAM roles are used to **delegate** access –
  - Used by an IAM user in the same AWS account as the role
  - Used by an AWS service—such as Amazon EC2—in the same account as the role
  - Used by an IAM user in a different AWS account than the role

IAM role

An **IAM role** is an IAM identity you can create in your account that has specific permissions. An IAM role is **similar to an IAM user** because it is also an AWS identity that you can attach permissions policies to, and those permissions determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, the role provides you with temporary security credentials for your role session.

You can **use roles to delegate access to users, applications, or services** that do not normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but you do not want to embed AWS keys within the app (where the keys can be difficult to rotate and where users can potentially extract them and misuse them). Also, sometimes you may want to grant AWS access to users who already have identities that are defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

For all of these example use cases, IAM roles are an essential component to implementing the cloud deployment.

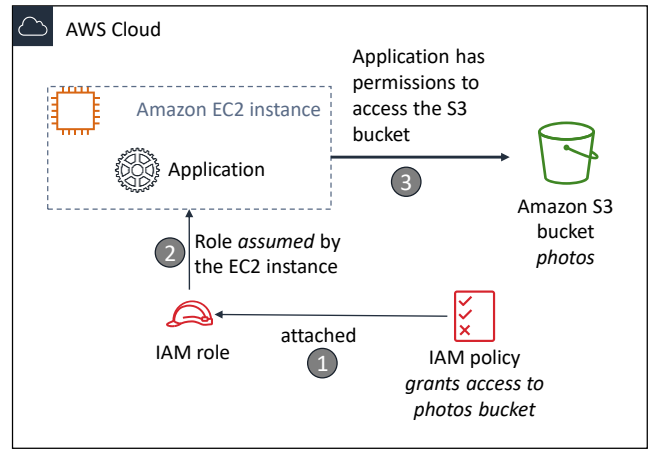
Example use of an IAM role

Scenario:

- An application that runs on an EC2 instance needs access to an S3 bucket

Solution:

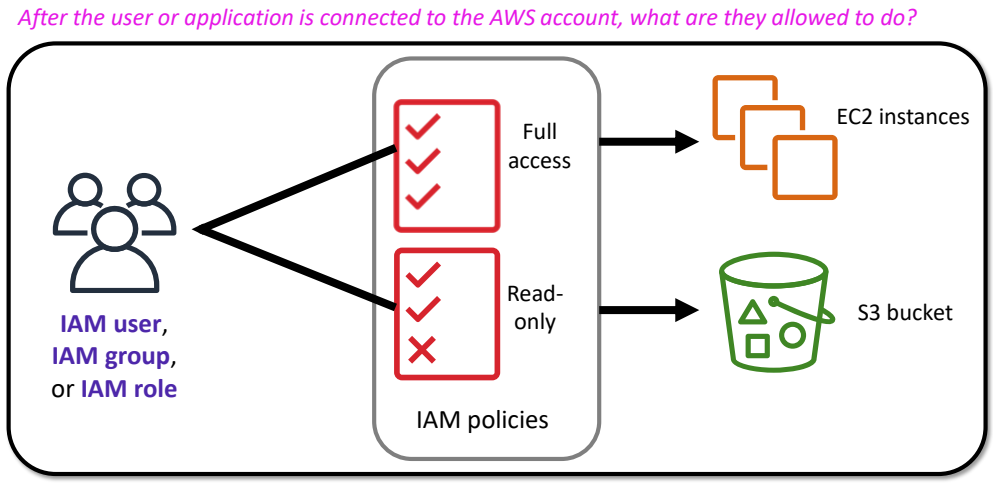
- Define an IAM policy that grants access to the S3 bucket.
- Attach the policy to a role
- Allow the EC2 instance to assume the role



In the diagram, a developer runs an application on an EC2 instance that requires access to the S3 bucket that is named *photos*. An administrator creates the IAM role and attaches the role to the EC2 instance. The role includes a permissions policy that grants read-only access to the specified S3 bucket. It also includes a trust policy that allows the EC2 instance to assume the role and retrieve the temporary credentials. When the application runs on the instance, it can use the role's temporary credentials to access the **photos** bucket. The administrator does not need to grant the application developer permission to access the photos bucket, and the developer never needs to share or manage credentials.

To learn more details about this example, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.

Authorization: What actions are permitted



Authorization is the process of determining what permissions a user, service or application should be granted. After a user has been authenticated, they must be authorized to access AWS services.

By default, IAM users do not have permissions to access any resources or data in an AWS account. Instead, you must explicitly grant permissions to a user, group, or role by creating a *policy*, which is a document in JavaScript Object Notation (JSON) format. A policy lists permissions that allow or deny access to resources in the AWS account.

IAM: Authorization

- Assign permissions by creating an IAM policy.
- Permissions determine **which resources and operations** are allowed:
  - All permissions are implicitly denied by default.
  - If something is explicitly denied, it is never allowed.

Best practice: Follow the **principle of least privilege**.

IAM permissions

Note: The scope of IAM service configurations is **global**. Settings apply across all AWS Regions.

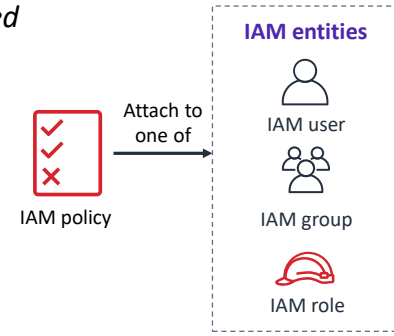
To assign permission to a user, group or role, you must create an **IAM policy** (or find an existing policy in the account). There are no default permissions. All actions in the account are denied to the user by default (*implicit deny*) unless those actions are explicitly allowed. Any actions that you do not explicitly allow are denied. Any actions that you explicitly deny are always denied.

The **principle of least privilege** is an important concept in computer security. It promotes that you grant only the minimal user privileges needed to the user, based on the needs of your users. When you create IAM policies, it is a best practice to follow this security advice of granting *least privilege*. Determine what users need to be able to do and then craft policies for them that let the users perform *only* those tasks. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too broad and then later trying to lock down the permissions granted.

Note that the scope of the IAM service configurations is **global**. The settings are not defined at an AWS Region level. IAM settings apply across all AWS Regions.

IAM policies

- **An IAM policy is a document that defines permissions**
  - Enables fine-grained access control
- Two types of policies – *identity-based* and *resource-based*
- **Identity-based** policies –
  - Attach a policy to any IAM entity
    - An IAM user, an IAM group, or an IAM role
  - Policies specify:
    - Actions that ***may*** be performed by the entity
    - Actions that ***may not*** be performed by the entity
  - A single *policy* can be attached to multiple *entities*
  - A single *entity* can have multiple *policies* attached to it
- **Resource-based** policies
  - Attached to a resource (such as an S3 bucket)



An IAM policy is a formal statement of permissions that will be granted to an entity. Policies can be attached to any IAM entity. Entities include users, groups, roles, or resources. For example, you can attach a policy to AWS resources that will block all requests that do not come from an approved Internet Protocol (IP) address range. Policies specify what actions are allowed, which resources to allow the actions on, and what the effect will be when the user requests access to the resources.

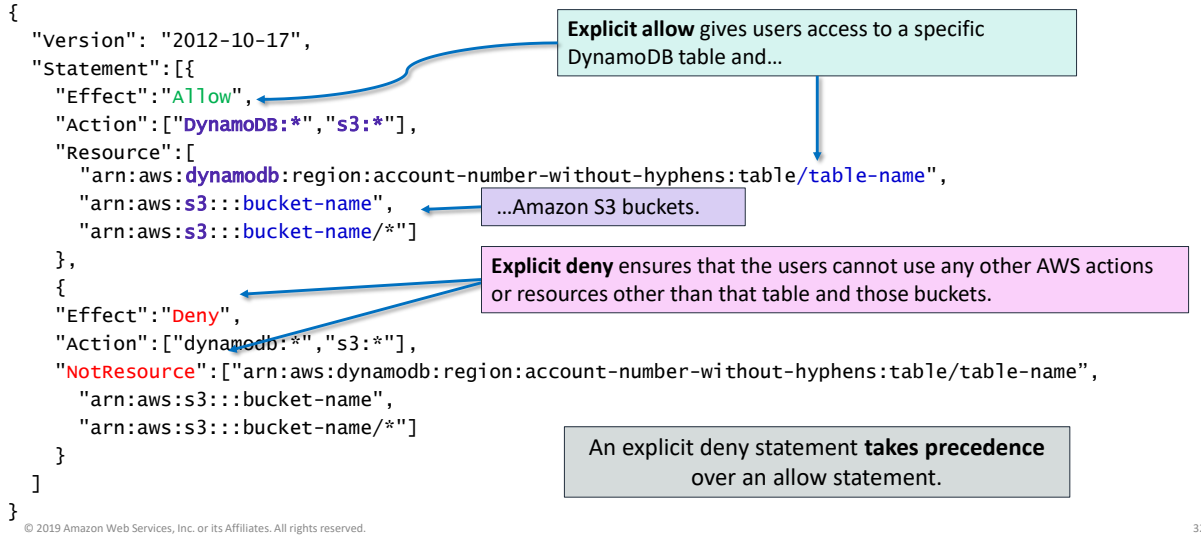
The order in which the policies are evaluated has no effect on the outcome of the evaluation. All policies are evaluated, and the result is always that the request is either allowed or denied. When there is a conflict, the most restrictive policy applies.

There are two types of IAM policies. **Identity-based policies** are permissions policies that you can attach to a principal (or identity) such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. Identity-based policies can be further categorized as:

- **Managed policies –** Standalone identity-based policies that you can attach to multiple users, groups, and roles in your AWS account
- **Inline policies –** Policies that you create and manage, and that are embedded directly into a single user group or role.

Resource-based policies are JSON policy documents that you attach to a resource, such as an S3 bucket. These policies control what actions a specified principal can perform on that resource, and under what conditions.

IAM policy example



As mentioned previously, IAM policy documents are written in JSON.

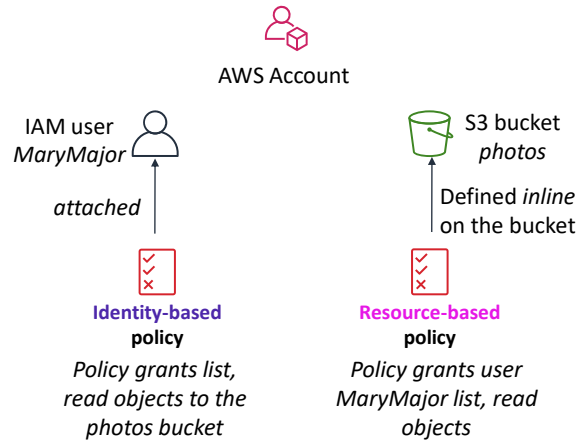
The example IAM policy grants users access only to the following resources:

- The DynamoDB table whose name is represented by *table-name*.
- The AWS account's S3 bucket, whose name is represented by *bucket-name* and all the objects that it contains.

The IAM policy also includes an explicit deny ("Effect":"Deny") element. The **NotResource** element helps to ensure that users cannot use any other DynamoDB or S3 actions or resources except the actions and resources that are specified in the policy—even if permissions have been granted in another policy. An explicit deny statement takes precedence over an allow statement.

Resource-based policies

- *Identity-based policies* are attached to a user, group, or role
- **Resource-based policies** are attached to a resource (*not* to a user, group or role)
- Characteristics of resource-based policies –
  - Specifies who has access to the resource and what actions they can perform on it
  - The policies are *inline* only, not managed
- Resource-based policies are supported only by some AWS services



While *identity-based policies* are attached to a user, group, or role, **resource-based policies** are attached to a resource, such as an S3 bucket. These policies specify who can access the resource and what actions they can perform on it.

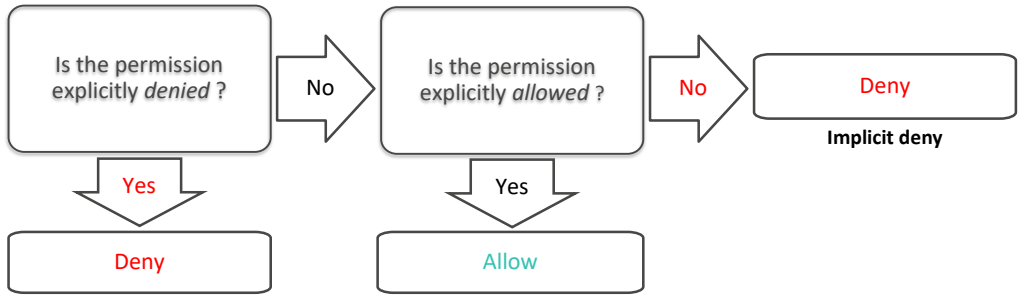
Resource-based policies are defined **inline** only, which means that you define the policy on the resource itself, instead of creating a separate IAM policy document that you attach. For example, to create an S3 bucket policy (a type of resource-based policy) on an S3 bucket, navigate to the bucket, click the **Permissions** tab, click the **Bucket Policy** button, and define the JSON-formatted policy document there. An Amazon S3 access control list (ACL) is another example of a resource-based policy.

The diagram shows two different ways that the user *MaryMajor* could be granted access to objects in the S3 bucket that is named *photos*. On the left, you see an example of an identity-based policy. An IAM policy that grants access to the S3 bucket is attached to the MaryMajor user. On the right, you see an example of a resource-based policy. The S3 bucket policy for the *photos* bucket specifies that the user MaryMajor is allowed to list and read the objects in the bucket.

Note that you could define a deny statement in a bucket policy to restrict access to specific IAM users, even if the users are granted access in a separate identity-based policy. An explicit deny statement will always take precedence over any allow statement.

IAM permissions

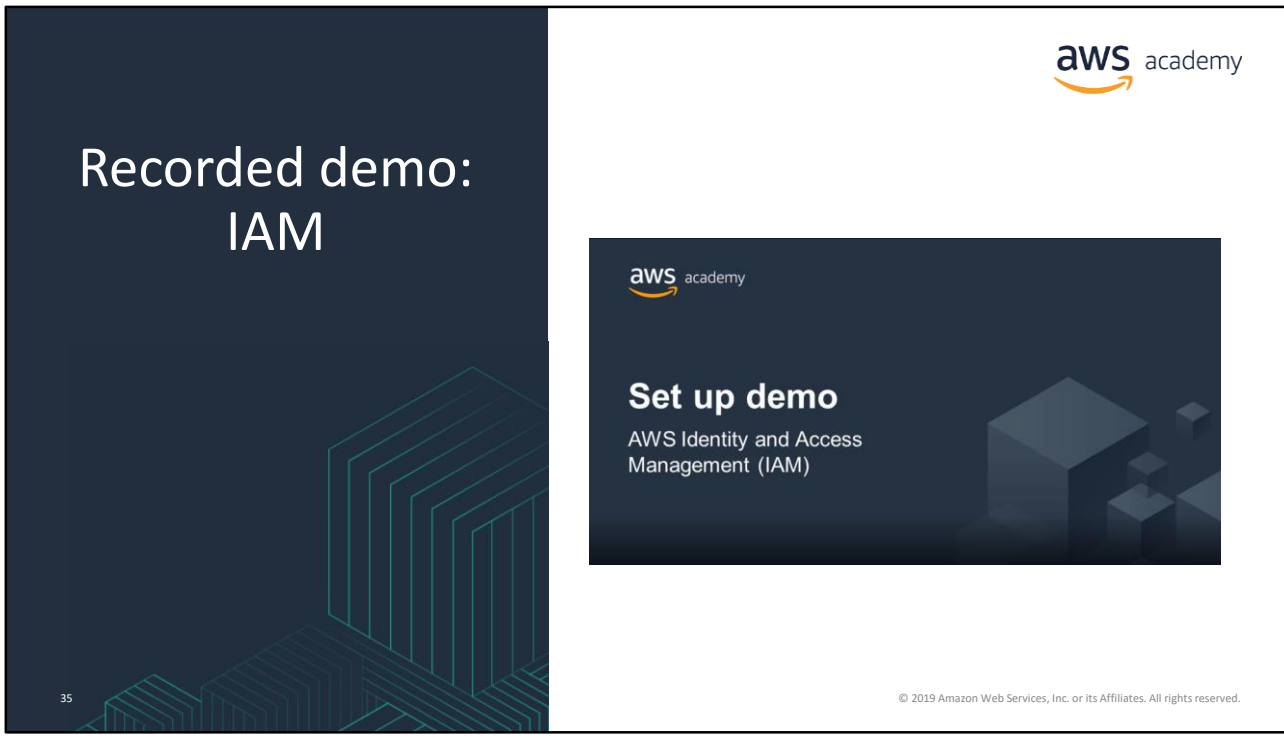
How IAM determines permissions:



IAM policies enable you to fine-tune privileges that are granted to IAM users, groups, and roles.

When IAM determines whether a permission is allowed, IAM first checks for the existence of any applicable **explicit denial policy**. If no explicit denial exists, it then checks for any applicable **explicit allow policy**. If neither an explicit deny nor an explicit allow policy exists, IAM reverts to the default, which is to deny access. This process is referred to as an **implicit deny**. The user will be permitted to take the action only if the requested action is *not* explicitly denied and *is* explicitly allowed.

It can be difficult to figure out whether access to a resource will be granted to an IAM entity when you develop IAM policies. The IAM Policy Simulator is a useful tool for testing and troubleshooting IAM policies.



Now, take a moment to watch the IAM Demo. The recording runs a little over 4 minutes, and it reinforces many of the concepts that were discussed in this section of the module.

The demonstration shows how to configure the following resources by using the AWS Management Console:

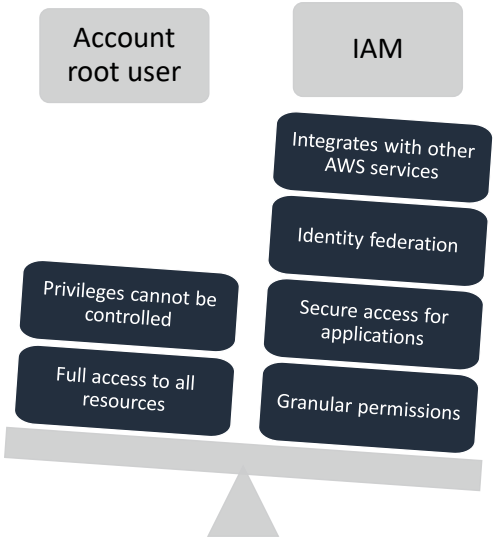
- An IAM role that will be used by an EC2 instance
- An IAM group
- An IAM user

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- **AWS IAM**
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - **Securing your AWS account**
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

AWS account root user access versus IAM access



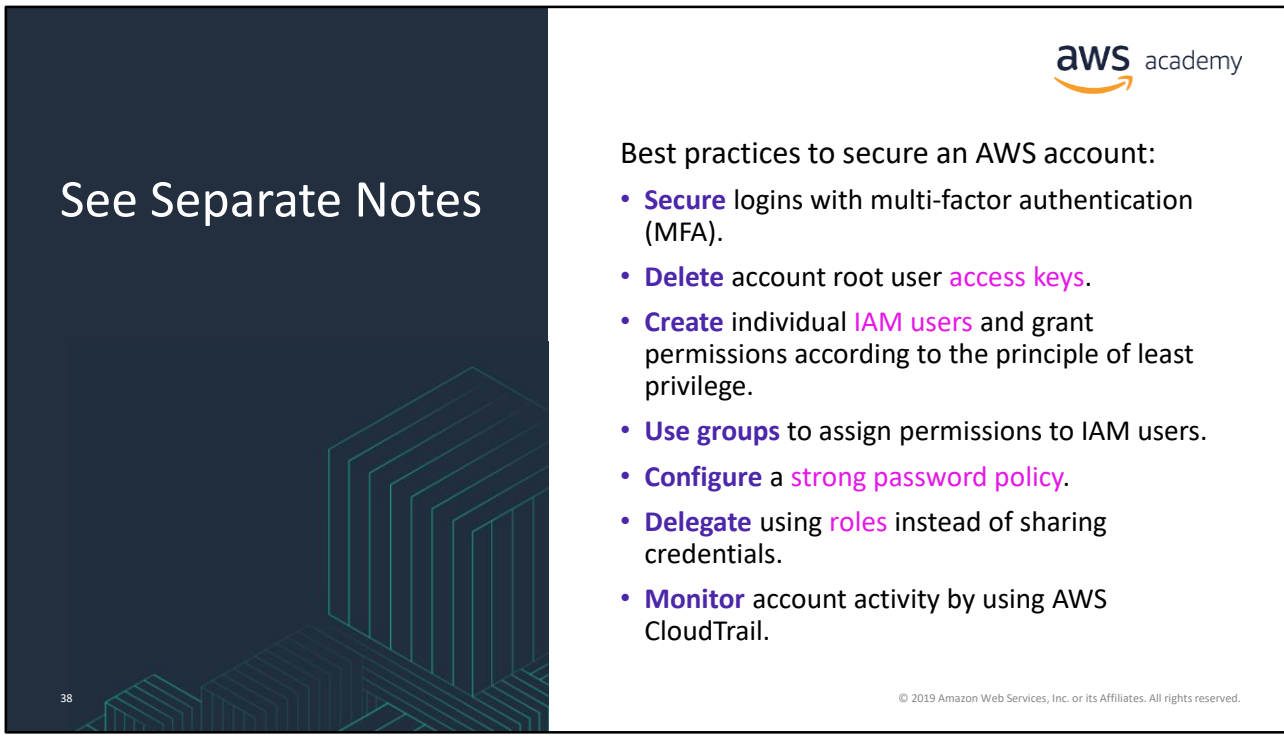
- **Best practice**: Do not use the AWS account root user except when necessary.
  - Access to the **account root user** requires logging in with the *email address* (and password) that you used to create the account.
- Example actions that can only be done with the account root user:
  - Update the account root user password
  - Change the AWS Support plan
  - Restore an IAM user's permissions
  - Change account settings (for example, contact information, allowed Regions)

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the **AWS account root user** and it is accessed by signing into the AWS Management Console with the email address and password that you used to create the account. AWS account root users have (and retain) ***full*** access to all resources in the account. Therefore, AWS strongly recommends that you do not use account root user credentials for day-to-day interactions with the account.

Instead, AWS recommends that you use IAM to create additional users and assign permissions to these users, following the principle of least privilege. For example, if you require administrator-level permissions, you can create an IAM user, grant that user full access, and then use those credentials to interact with the account. Later, if you need to revoke or modify your permissions, you can delete or modify any policies that are associated with that IAM user.

Additionally, if you have multiple users that require access to the account, you can create unique credentials for each user and define which user will have access to which resources. For example, you can create IAM users with read-only access to resources in your AWS account and distribute those credentials to users that require read access. You should avoid sharing the same credentials with multiple users.

While the account root user should not be used for routine tasks, there are a few tasks that can only be accomplished by logging in as the account root user. A full list of these tasks is detailed on the AWS Tasks that Require AWS Account Root User Credentials AWS documentation page.



The key takeaways from this section of the module are all related to best practices for securing an AWS account. Those best practice recommendations include:

- Secure logins with multi-factor authentication (MFA).
- Delete account root user access keys.
- Create individual IAM users and grant permissions according to the principle of least privilege.
- Use groups to assign permissions to IAM users.
- Configure a strong password policy.
- Delegate using roles instead of sharing credentials.
- Monitor account activity using AWS CloudTrail.

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- **AWS Authentication**
  - **Cognito**, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

Amazon Cognito

Amazon Cognito features:

- **Adds user sign-up, sign-in, and access control to your web and mobile applications**.
- Scales to millions of users.
- Supports sign-in with social identity providers, such as Facebook, Google, and Amazon; and enterprise identity providers, such as Microsoft Active Directory via Security Assertion Markup Language (SAML) 2.0.

Amazon Cognito

Amazon Cognito provides solutions to control access to AWS resources from your application. You can define roles and map users to different roles so your application can access only the resources that are authorized for each user.

Amazon Cognito uses common identity management standards, such as **Security Assertion Markup Language (SAML) 2.0**. SAML is an open standard for exchanging identity and security information with applications and service providers. Applications and service providers that support SAML enable you to sign in by using your corporate directory credentials, such as your user name and password from Microsoft Active Directory. With SAML, you can use single sign-on (SSO) to sign in to all of your SAML-enabled applications by using a single set of credentials.

Amazon Cognito helps you **meet multiple security and compliance requirements**, including requirements for highly regulated organizations such as healthcare companies and merchants. Amazon Cognito is eligible for use with the US Health Insurance Portability and Accountability Act (HIPAA). It can also be used for workloads that are compliant with the Payment Card Industry Data Security Standard (PCI DSS); the American Institute of CPAs (AICPA) Service Organization Control (SOC); the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) standards ISO/IEC 27001, ISO/IEC 27017, and ISO/IEC 27018; and ISO 9001.

Amazon Cognito

Amazon Cognito

- Provides authentication, authorization, and user management for your web and mobile applications
- Authenticates user identities through external IdPs that support SAML or OpenID Connect, social IdPs, and custom IdPs
- Synchronizes data across a user's devices and saves data locally
- Provides temporary security credentials to access AWS resources or any service behind Amazon API Gateway

Amazon Cognito is a service that provides authentication, authorization, and user management for your web and mobile applications. You can use Amazon Cognito to:

- Create unique identities for your users and authenticate the identities with identity providers. Amazon Cognito works with external identity providers that support SAML or OpenID Connect, social identity providers (such as Facebook, Twitter, and Amazon), and your own identity provider.
- Synchronize data across a user's devices so their application experience remains consistent when they switch between devices or upgrade to a new device. Amazon Cognito can also save data locally on users' devices, which allows your application to work when devices are offline. Amazon Cognito then automatically synchronizes data when the devices are back online.
- Provide temporary security credentials to your application to access AWS resources or any service behind Amazon API Gateway.

For more information about Amazon Cognito, see the following resources:

- Amazon Cognito: https://aws.amazon.com/cognito/
- Amazon Cognito FAQs: https://aws.amazon.com/cognito/faqs/
- What Is Amazon Cognito?: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Amazon Cognito components

aws academy

User pool
(Authenticates users)

- User directory in Amazon Cognito
- Provides sign-up and sign-in options for your application users
- Users can sign in to your application through Amazon Cognito or federate through a third-party IdP.

Identity pool
(Grants users access)

- Provides your users temporary security credentials access to other AWS services

The two main components of Amazon Cognito are user pools and identity pools.

- *User pools* are user directories that provide sign-up and sign-in options for your application users. Users can sign in to your web or mobile application through Amazon Cognito, or federate through a third-party identity provider. 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.
- *Identity pools* provide your users temporary security credentials to access other AWS services.

As you will see, you can use identity pools and user pools separately or together.

See the AWS documentation for more information:

- Amazon Cognito user pools and identity pools: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html
- Amazon Cognito user pools: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html
- Amazon Cognito identity pools: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-identity.html
- Common Amazon Cognito Scenarios:

https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-scenarios.html

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- **AWS Authentication**
  - Cognito, **Directory service**, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

AWS Directory Service

- AWS Directory Service is a managed service to:
  - Run Microsoft AD as a managed service within AWS Directory Service
  - Connect your AWS resources with an existing on-premises Microsoft Active Directory (AD Connector)
  - Set up a new, stand-alone directory in the AWS Cloud (Simple AD)
- AWS Directory Service allows use of existing corporate credentials for:
  - Accessing AWS services (e.g. Amazon WorkSpaces, and Amazon WorkDocs)
  - Accessing the AWS Management Console through IAM Roles
- Options:
  - Run Microsoft AD as a managed service within AWS Directory Service
  - Use AD Connector
  - Use Simple AD

AWS Directory Service is a managed service that allows you to connect your AWS resources with an existing on-premises Microsoft Active Directory or to set up a new, stand-alone directory in the AWS Cloud.

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- **AWS Authentication**
  - Cognito, Directory service, **STS**, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

Why use temporary security credentials

Bad practice

- Embed access keys in unencrypted code
- Share access keys between users in AWS account

Best practice

- Use IAM roles to retrieve temporary security credentials

As you learned in the IAM module, you can authenticate programmatically to AWS services through the AWS Command Line Interface (AWS CLI), software development kits (SDKs), and APIs using your AWS access key. The *access key* is a combination of your access key ID and secret access key.

Embedding access keys in unencrypted code and sharing security credentials between users in your AWS account are bad security practices. If your application or users of your application need to access AWS services, you should configure *temporary security credentials*.

For more information about best practices around temporary security credentials, see IAM Best Practices in the AWS documentation: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Temporary security credentials

AWS Security Token Service

- Provided by AWS Security Token Service to trusted users to enable them to access your AWS resources
- Short-lived access key ID, secret access key, and a session token
- Limited, configurable lifetime
- Cannot be reused after they expire
- Generated dynamically

You can use AWS Security Token Service (AWS STS) to provide trusted users with temporary security credentials to access your AWS resources. Temporary security credentials consist of a short-lived access key ID, a secret access key, and a session token.

As the name implies, temporary security credentials have a limited lifetime. They can be configured to last from a few minutes to several hours. After the credentials expire, AWS no longer recognizes them or allows any kind of access from API requests made with them.

Temporary security credentials are not stored with the user but are generated dynamically and provided to the user when requested. When (or even before) the temporary security credentials expire, the user can request new credentials, as long as the user requesting them still has permissions to do so.

See the AWS documentation for more information:

- AWS Security Token Service: https://docs.aws.amazon.com/STS/latest/APIReference/Welcome.html

- Temporary Security Credentials: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html
- List of IAM features that AWS STS supports: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html

AWS STS trusted users

- IAM users
  - For cross-account access
  - For applications running on Amazon Elastic Compute Cloud (Amazon EC2) instances and other AWS compute services
- Federated identities for external users
  - Enterprise (single sign-on)
  - Web (social sign-in)

AWS STS trusted users can be:

- **IAM users** – You can establish cross-account access for IAM users in one AWS account who need temporary access to AWS resources in another AWS account. To do this, you can provide temporary security credentials to an IAM role that the IAM user assumes. You can also use IAM roles with temporary security credentials to manage access to applications running in an Amazon Elastic Compute Cloud (Amazon EC2) environment and other AWS compute services.
- **Federated identities** – *Federated identities* are users who sign in to your application from an authentication system outside of AWS. IAM supports two types of identity federation:
  - **Enterprise identity federation** – For employees who are on a corporate network and are authenticated via a Security Assertion Markup Language (SAML) 2.0-compatible enterprise identity provider, such as Microsoft Active Directory, Lightweight Directory Access Protocol (LDAP), and so forth.
  - **Web identity federation** – For mobile and web-based application users who are authenticated via an online third-party identity provider, such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)-compatible identity provider.

For federated identities, you do not need to create new AWS identities for users and require them to sign in to your application with a separate user name and password.

Instead, users can access your AWS resources directly using their corporate network credentials (referred to as single sign-on, or SSO) or through a third party, such as Login with Amazon, Facebook, or Google (referred to as *social sign-in*).

See the AWS documentation for more information:

- Identity Providers and Federation: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers.html
- Providing access to externally authenticated users: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_common-scenarios_federated-users.html#id_roles_common-scenarios_federated-users-idbroker

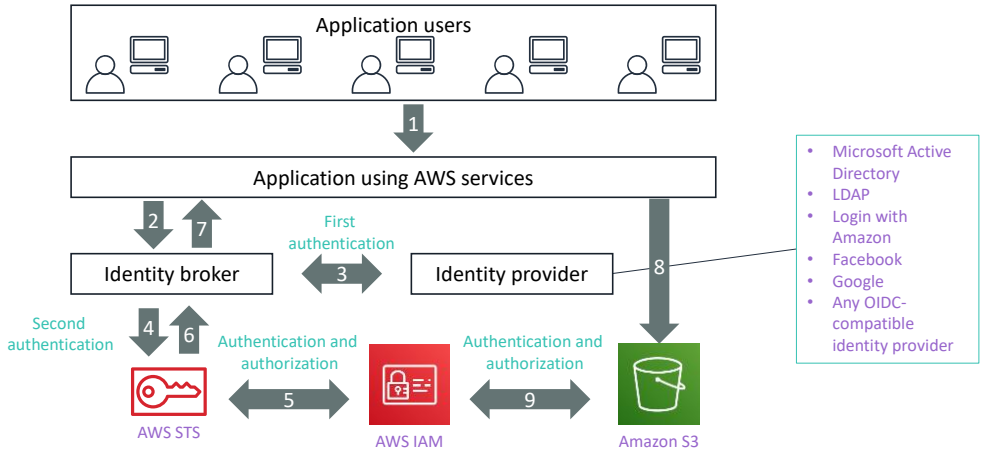
AWS STS key concepts

- **Authentication** – Verify identity of user
- **Authorization** – Verify permissions (what the user is allowed to do)
- **Identity provider (IdP)** – Manages identity information and provides authentication services
- **Identity broker** – Authenticates credentials against an IdP and retrieves temporary security credentials from AWS STS
- **Standards:**
  - **SAML (Security Assertion Markup Language)** – Open standard used for exchanging authentication and authorization data between parties
  - **OpenID Connect (OIDC)** – Open standard used by third-party IdPs so other companies/sites can use them to authenticate users without having to maintain an in-house user database

To understand how AWS STS works, you should familiarize yourself with the following key concepts:

- **Authentication** – Verifies user identity
- **Authorization** – Verifies the user's permissions (or what the user is allowed to do)
- **Identity provider (IdP)** – Manages identity information and provides authentication services
- **Identity broker** – Software layer that authenticates credentials against an IdP and retrieves temporary security credentials from AWS STS
- **Secure Assertion Markup Language (SAML)** – Open standard used to exchange authentication and authorization data between parties
- **OpenID Connect (OIDC)** – Open standard used by third-party IdPs so other companies or sites can use them to authenticate users without having to maintain an in-house user database

Authentication with AWS STS



Here's how authentication with AWS STS works.

1. A user uses an application backed by AWS.
2. The application calls an *identity broker*. The identity broker accepts a user's identifier as input.
3. (First authentication) The identity broker first authenticates the user's identity against an *identity provider* (IdP) such as Microsoft Active Directory (for enterprise federation), an online third-party IdP (for web federation), or against IAM (for IAM users).
4. (Second authentication) If the authentication is successful, the identity broker makes an API call to AWS STS. The call must include an IAM policy and a duration, along with a policy that specifies the permissions to be granted to the temporary security credentials.
5. AWS STS uses IAM to confirm that the policy of the IAM user that is making the API call has permission to create new tokens.
6. AWS STS returns four values to the identity broker—an access key, a secret access key, a session token, and a duration (that is, the token's lifetime).
7. The identity broker returns the temporary security credentials and token to the application.
8. The application uses the temporary security credentials and token to make requests to an AWS service, such as Amazon S3.
9. The AWS service uses IAM to confirm that the credentials allow the requested

operation on the given resource.

For more information about how authentication with AWS STS works, see the following resources:

- AWS Identity and Access Management – Now With Identity Federation: https://aws.amazon.com/blogs/aws/aws-identity-and-access-management-now-with-identity-federation/
- Requesting Temporary Security Credentials: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_request.html

AWS STS API operations

Operation	Returns Set of Temporary Security Credentials to…
`AssumeRole`	Existing IAM users for cross-account access to AWS resources.
`AssumeRoleWithSAML`	Federated users who are authenticated by an organization's existing identity system.
`AssumeRoleWithWebIdentity`	Federated users who are authenticated through a public identity provider.
`GetFederationToken`	Federated users. This API differs from `AssumeRole` in that the default expiration period is substantially longer (12 hours instead of 1 hour).
`GetSessionToken`	Existing IAM users for enhanced security, such as allowing AWS requests only when MFA is enabled for the IAM user.

Different AWS STS API operations return temporary security credentials:

- **AssumeRole** – Returns a set of temporary security credentials for existing IAM users to grant cross-account access to AWS resources.
- **AssumeRoleWithSAML** – Returns a set of temporary security credentials for federated users who are authenticated by an organization's existing identity system. The users must also use SAML 2.0 to pass authentication and authorization information to AWS.
- **AssumeRoleWithWebIdentity** – Returns a set of temporary security credentials for federated users who are authenticated through a public identity provider, such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC)-compatible identity provider. This API is useful for creating mobile applications or client-based web applications that require access to AWS in which users do not have their own AWS or IAM identities. However, instead of directly calling AssumeRoleWithWebIdentity, we recommend that you use Amazon Cognito and the Amazon Cognito credentials provider with the AWS SDKs for mobile development.
- **GetFederationToken** – Returns a set of temporary security credentials for federated users. This API differs from AssumeRole in that the default expiration period is substantially longer (12 hours instead of 1 hour).
- **GetSessionToken** – Returns a set of temporary security credentials for an existing IAM user. This is useful for providing enhanced security, such as allowing AWS requests only when multi-factor authentication (MFA) is enabled for the IAM user.

See the AWS documentation for more information:

- AWS Security Token Service API operations for requesting temporary security credentials: https://docs.aws.amazon.com/STS/latest/APIReference/Welcome.html
- Comparing the AWS STS API operations: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_request.html#stsapi_comparison

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- **AWS Authentication**
  - Cognito, Directory service, STS, **Web Identity**
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

Web Identity Federation

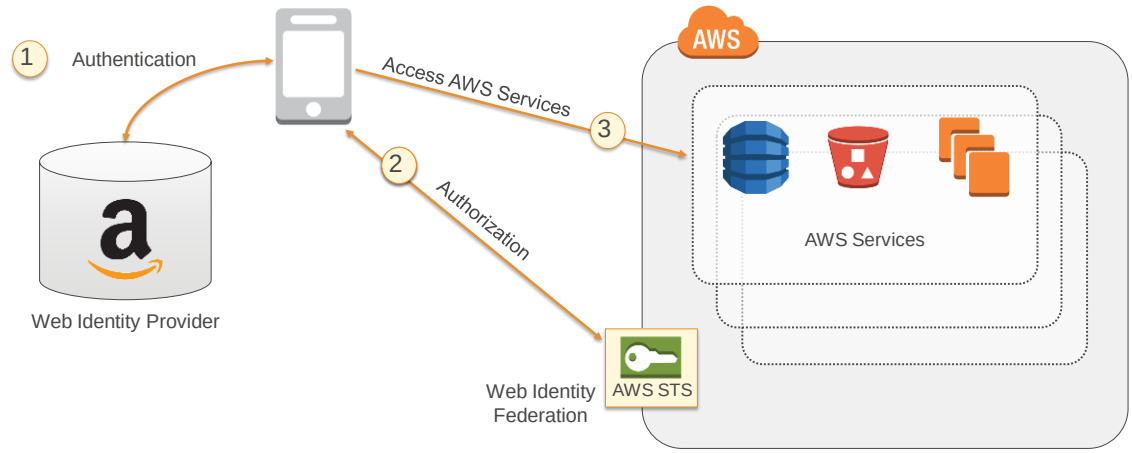
- Use STS API, **`AssumeRoleWithWebIdentity`**
  - Lets you request temporary security credentials to access AWS resources.
- Supported web identity providers:
  - Amazon
  - Google
  - Facebook
- A mobile app can be developed without server-side code and without distributing long-term credentials with the mobile app.

To configure your application and vend federated credentials using trusted identity providers such as Login with Amazon, Facebook, or Google, you want to:

1. Register an application with the identity provider.
2. Create an IAM role for the identity provider.
3. Set up permissions for the IAM role.
4. Use the identity provider's SDK to get an access to token after logging in.
5. Use the AWS SDK for JavaScript to get temporary credentials to your application.

For more information, see the online documentation at http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WIF.RunningYourApp.html

Use Case: Web Identity Federation



1. The user needs to be authenticated. For example, using the *Login with Amazon* SDK, the app authenticates the user and receives a token from Amazon.
2. The user needs to be authorized to access resources in his/her AWS account. The app makes an unsigned AssumeRoleWithWebIdentity request to STS, passing the token from the previous step. STS verifies the authenticity of the token; if the token is valid, STS returns a set of temporary security credentials to the app. By default, the credentials can be used for one hour.
3. The app uses the temporary security credentials to make signed requests for resources in your Amazon S3 bucket (as an example). Because the role's access policy used variables that reference the app ID and the user ID, the temporary security credentials are scoped to that end user and will prevent him/her from accessing objects owned by other users.

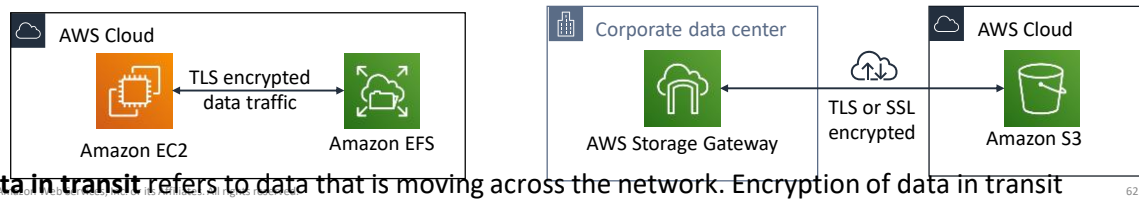
This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- **Securing Data**
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

Encryption of data *in transit*

- Encryption of **data in transit** (data moving across a network)
  - **Transport Layer Security (TLS)**—formerly SSL—is an open standard protocol
  - **AWS Certificate Manager** provides a way to manage, deploy, and renew TLS or SSL certificates
- Secure HTTP (HTTPS) creates a secure tunnel
  - Uses TLS or SSL for the bidirectional exchange of data
- **AWS services support data in transit encryption**.
  - Two examples:



Data in transit refers to data that is moving across the network. Encryption of data in transit is accomplished by using Transport Layer Security (TLS) 1.2 with an open standard AES-256 cipher. TLS was formerly called Secure Sockets Layer (SSL).

AWS Certificate Manager is a service that enables you to provision, manage, and deploy SSL or TLS certificates for use with AWS services and your internal connected resources. SSL or TLS certificates are used to secure network communications and establish the identity of websites over the internet, and also resources on private networks. With AWS Certificate Manager, you can request a certificate and then deploy it on AWS resources (such as load balancers or CloudFront distributions). AWS Certificate Manager also handles certificate renewals.

Web traffic that runs over HTTP is not secure. However, traffic that runs over **Secure HTTP (HTTPS)** is encrypted by using TLS or SSL. HTTPS traffic is protected against eavesdropping and man-in-the-middle attacks because of the bidirectional encryption of the communication.

AWS services support encryption for data in transit. Two examples of encryption for data in transit are shown. The first example shows an EC2 instance that has mounted an Amazon EFS shared file system. All data traffic between the instance and Amazon EFS is encrypted by using TLS or SSL. For further details about this configuration, see Encryption of EFS Data in Transit.

The second example shows the use of **AWS Storage Gateway,** a hybrid cloud storage service that provides on-premises access to AWS Cloud storage. In this example, the storage gateway is connected across the internet to Amazon S3, and the connection encrypts the data in transit.

Encryption of data *at rest*

- **Encryption** encodes data with a secret key, which makes it unreadable
  - Only those who have the secret key can decode the data
  - AWS KMS can manage your secret keys

- AWS supports encryption of **data at rest**
  - Data at rest = Data stored physically (on disk or on tape)
  - You can encrypt data stored in any service that is supported by AWS KMS, including:
    - Amazon S3
    - Amazon EBS
    - Amazon Elastic File System (Amazon EFS)
    - Amazon RDS managed databases

Data encryption is an essential tool to use when your objective is to protect digital data. Data encryption takes data that is legible and encodes it so that it is unreadable to anyone who does not have access to the secret key that can be used to decode it. Thus, even if an attacker gains access to your data, they cannot make sense of it.

Data at rest refers to data that is physically stored on disk or on tape.

You can create encrypted file systems on AWS so that all your data and metadata is encrypted at rest by using the open standard Advanced Encryption Standard (AES)-256 encryption algorithm. When you use AWS KMS, encryption and decryption are handled automatically and transparently, so that you do not need to modify your applications. If your organization is subject to corporate or regulatory policies that require encryption of data and metadata at rest, AWS recommends enabling encryption on all services that store your data. You can encrypt data stored in any service that is supported by AWS KMS. See How AWS Services use AWS KMS for a list of supported services.

Securing Amazon S3 buckets and objects

- Newly created S3 buckets and objects are private and protected by default.
- When use cases require sharing data objects on Amazon S3 –
  - It is essential to manage and control the data access.
  - Follow the **permissions that follow the principle of least privilege** and consider using Amazon S3 encryption.
- Tools and options for controlling access to S3 data include –
  - Amazon S3 Block Public Access feature: Simple to use.
  - IAM policies: A good option when the user can authenticate using IAM.
  - Bucket policies
  - Access control lists (ACLs): A legacy access control mechanism.
  - AWS Trusted Advisor bucket permission check: A free feature.

By default, all Amazon S3 buckets are private and can be accessed *only* by users who are explicitly granted access. It is essential to manage and control access to Amazon S3 data.
AWS provides many tools and options for controlling access to your S3 buckets or objects, including:

- Using **Amazon S3 Block Public Access**. These settings override any other policies or object permissions. Enable **Block Public Access** for all buckets that you don't want to be publicly accessible. This feature provides a straightforward method for avoiding unintended exposure of Amazon S3 data.

- Writing **IAM policies** that specify the users or roles that can access specific buckets and objects. This method was discussed in detail earlier in this module.

- Writing **bucket policies** that define access to specific buckets or objects. This option is typically used when the user or system cannot authenticate by using IAM. Bucket policies can be configured to grant access across AWS accounts or to grant public or anonymous access to Amazon S3 data. If bucket policies are used, they should be written carefully and tested fully. You can specify a deny statement in a bucket policy to restrict access. Access will be restricted even if the users have permissions that are granted in an identity-based policy that is attached to the users.

- Setting **access control lists (ACLs)** on your buckets and objects. ACLs are less commonly used (ACLs predate IAM). If you do use ACLs, do not set access that is too open or permissive.

- **AWS Trusted Advisor** provides a bucket permission check feature that is a useful tool for discovering if any of the buckets in your account have permissions that grant global access.

AWS Key Management Service (AWS KMS)

AWS Key Management Service (AWS KMS) features:

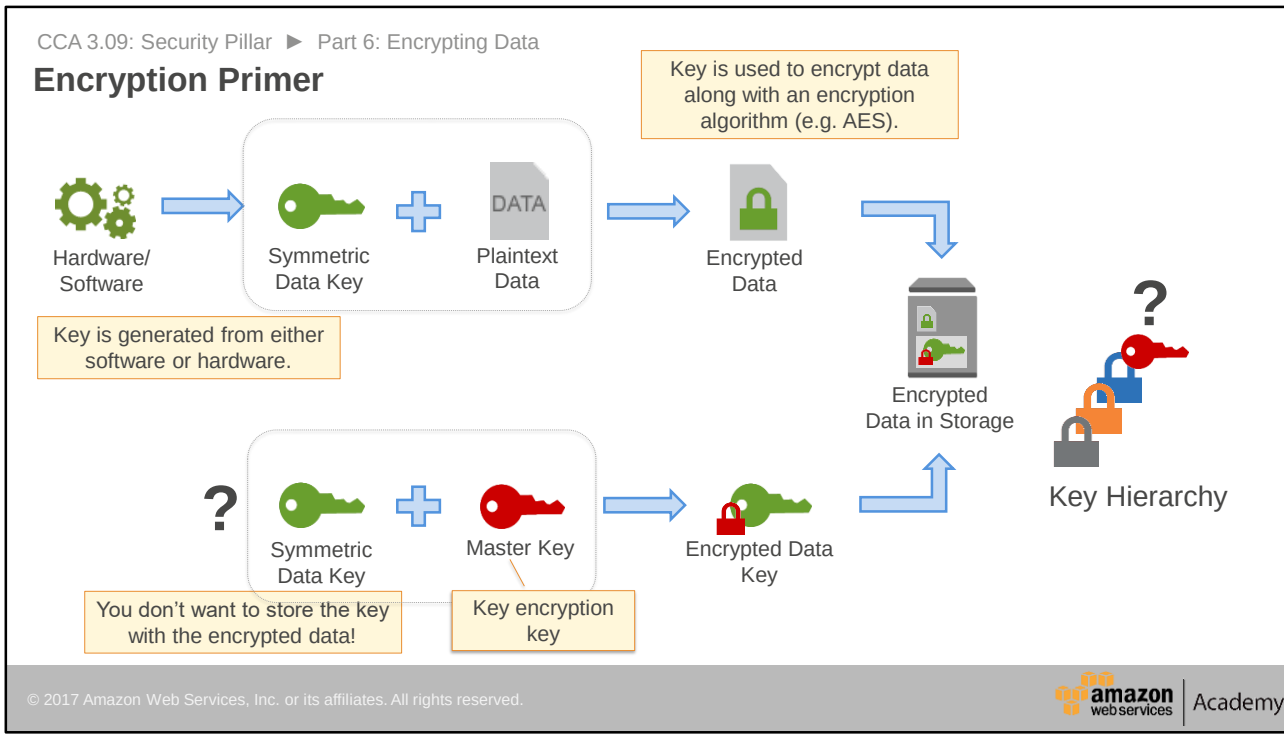
- Enables you to **create and manage encryption keys**
- Enables you to control the use of encryption across AWS services and in your applications.
- Integrates with AWS CloudTrail to log all key usage.
- Uses hardware security modules (HSMs) that are validated by Federal Information Processing Standards (FIPS) 140-2 to protect keys

AWS Key Management Service (AWS KMS)

AWS Key Management Service (AWS KMS) is a service that enables you to create and manage encryption keys, and to control the use of encryption across a wide range of AWS services and your applications. AWS KMS is a secure and resilient service that uses hardware security modules (HSMs) that were validated under **Federal Information Processing Standards (FIPS) 140-2** (or are in the process of being validated) to protect your keys. AWS KMS also integrates with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs.

Customer master keys (CMKs) are used to control access to data encryption keys that encrypt and decrypt your data. You can create new master keys when you want, and you can manage who has access to these keys and which services they can be used with. You can also import keys from your own key management infrastructure into AWS KMS.

AWS KMS integrates with most AWS services, which means that you can use AWS KMS master keys to control the encryption of the data that you store in these services. To learn more, see AWS Key Management Service features.



Before we discuss specific encryption and key management functions in AWS, let's review how data encryption and key management is typically implemented.

A symmetric data key is generated from either software or hardware. Symmetric keys are preferable to asymmetric keys when you want to encrypt data of an arbitrary size and have it be fast.

The key is used along with an encryption algorithm (like AES), and the resulting ciphertext is stored.

But what about the symmetric key you just used? You can't store it with the encrypted data. You have to protect that key somehow.

The best practice is to encrypt the data key with yet another key, called a key-encrypting key. This key can be symmetric or asymmetric, but it should be derived and stored in a separate system than the one you're processing your data in.

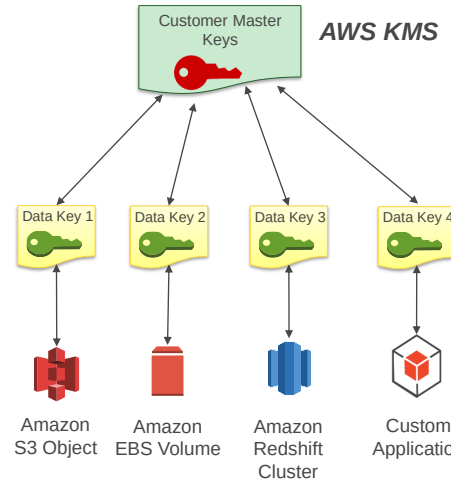
After you encrypt the data key with the key-encrypting key, you can then store the resulting ciphertext along with the encrypted data.

But what about the key-encrypting key? How do you protect that? You can iterate on the process of enveloping this key with additional keys as many times as you want, creating a key hierarchy. At some point, you'll need to access a plaintext key that starts the "unwrapping" process so you can derive the final data key to decrypt the data. The location and access controls around this key should be distinct from the ones used with the original data.

What Is AWS Key Management Service (KMS)?

AWS KMS is a managed encryption service that enables you to easily encrypt your data.

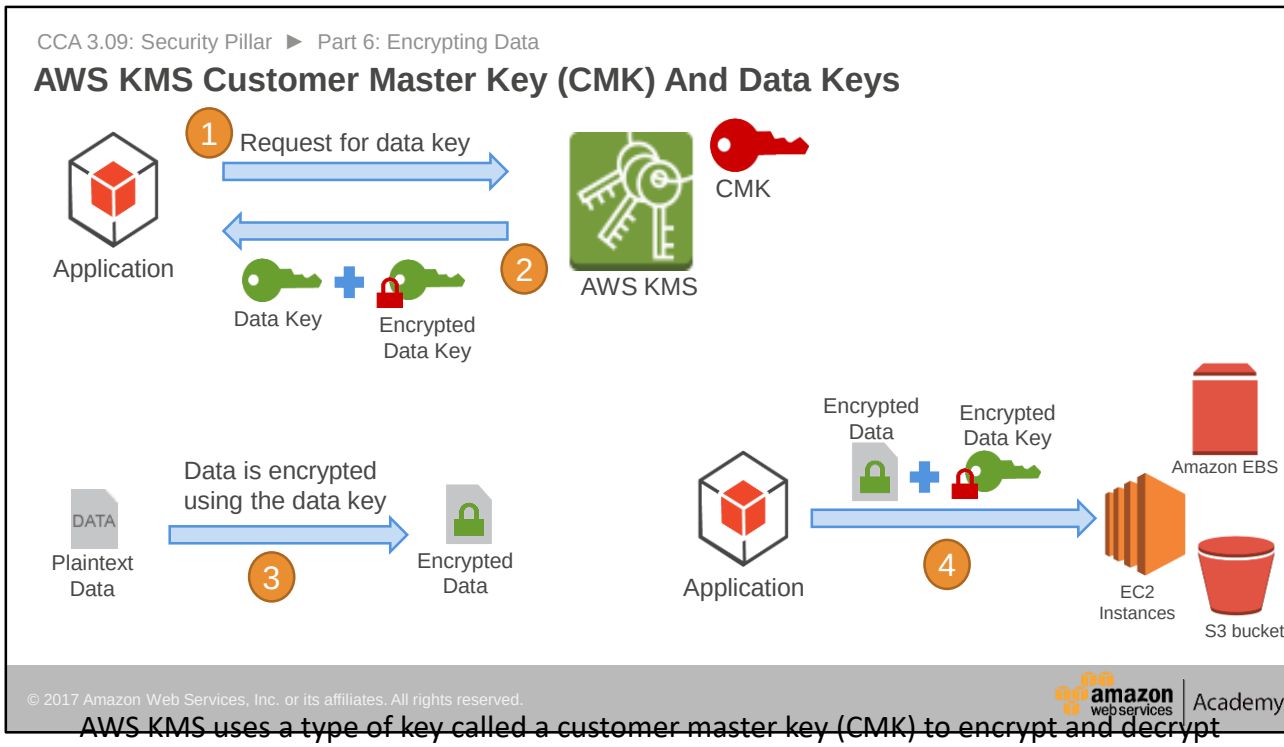
- Two-tiered key hierarchy using envelope encryption.
- Data keys are unique.
- AWS KMS master keys encrypt data keys.
- AWS KMS master keys never leave the AWS KMS system.



AWS KMS is a managed encryption service that enables you to easily encrypt your data. AWS KMS provides a highly available key storage, management, and auditing solution for you to encrypt your data across AWS services and within your own applications.

If you are a developer who needs to encrypt data in your applications, use the AWS SDKs with AWS KMS support to easily use and protect encryption keys. If you're an IT administrator looking for a scalable key management infrastructure to support your developers and their growing number of applications, use AWS KMS to reduce your licensing costs and operational burden. If you're responsible for providing data security for regulatory or compliance purposes, use AWS KMS to verify that data is encrypted consistently across the applications where it is used and stored.

AWS KMS uses envelope encryption to protect data. AWS KMS creates a data key, encrypts it under a customer master key, and returns plaintext and encrypted versions of the data key to you. You use the plaintext key to encrypt data and store the encrypted key alongside the encrypted data. The key should be removed from memory as soon as is practical after use. You can retrieve a plaintext data key only if you have the encrypted data key and you have permission to use the corresponding master key.



AWS KMS uses a type of key called a customer master key (CMK) to encrypt and decrypt data. CMKs are the fundamental resources that AWS KMS manages. CMKs can be either customer-managed keys or AWS-managed keys. They can be used inside of AWS KMS to encrypt or decrypt up to 4 kilobytes of data directly. They can also be used to encrypt generated data keys, which are then used to encrypt or decrypt larger amounts of data outside of the service. CMKs can never leave AWS KMS unencrypted, but data keys can. There is one AWS-managed key for each account for each service that is integrated with AWS KMS. This key is referred to as the *default* key for the service under your account. This key is used to encrypt data keys used by AWS services to protect data when you don't specify a CMK while creating the encrypted resource. If you need more granular control, you can specify a customer-managed key. For example, if you choose to encrypt an Amazon EBS volume, you can specify the AWS-managed default EBS key for the account or a CMK you created within AWS KMS. The key you selected is then used to protect the data key used to encrypt the volume.

You can only create CMKs if you have the appropriate permissions. You can provide an alias (display name) and a description for the key and define which IAM users or roles within an account can manage and use the key. You can also choose to allow AWS accounts other than your own to use the key. You use data keys to encrypt large data objects within your own application outside AWS KMS. When you call **GenerateDataKey**, AWS KMS returns a plaintext version of the key and cipher text that contains the key encrypted under the specified CMK. AWS KMS tracks which CMK was used to encrypt the data key. You use the

plaintext data key in your application to encrypt data, and you typically store the encrypted key alongside your encrypted data. Security best practices suggest that you should remove the plaintext key from memory as soon as is practical after use. To decrypt data in your application, pass the encrypted data key to the Decrypt function. AWS KMS uses the associated CMK to decrypt and retrieve your plaintext data key. Use the plaintext key to decrypt your data and then remove the key from memory.

Benefits Of Using AWS KMS

- Only data keys are available directly to the customer, and these are unique to each item encrypted. If one was compromised, it would not allow decryption of other objects.
- The risk of a compromised data key is limited.
- The performance for encrypting large data is improved.
- It is easier to manage a small number of master keys than millions of data keys.

AWS KMS allows you to centrally manage and securely store your keys. These keys can be used from within your applications and supported AWS cloud services to protect your data, but the key never leaves AWS KMS. You submit data to AWS KMS to be encrypted or decrypted under keys that you control. You set usage policies on these keys that determine which users can use them to encrypt and decrypt data. All requests to use these keys are logged in AWS CloudTrail so you can understand who used which key when.

You can perform the following key management functions in AWS KMS:

- Create keys with a unique alias and description.
- Define which IAM users and roles can manage keys.
- Define which IAM users and roles can use keys to encrypt and decrypt data.
- Choose to have AWS KMS automatically rotate your keys on an annual basis.
- Temporarily disable keys so they cannot be used by anyone.
- Re-enable disabled keys.
- Audit use of keys by inspecting logs in AWS CloudTrail.

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- **Securing the application**
  - **DDOS**, AWS WAF
- Auditing
  - AWS CloudTrail, Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

DDoS (Distributed Denial of Service) Attacks

- A Denial of Service (DoS) attack attempts to make your website or application unavailable to your end users.
- To achieve this, attackers use a variety of techniques that consume network or other resources, thus interrupting access for legitimate end users.
- The attackers use multiple hosts to orchestrate an attack against a target.

This section gives an overview of Distributed Denial of Service (DDoS) attacks and discusses techniques using AWS services as well as security solutions from the AWS Marketplace to help build resilience into your architecture. We will discuss anti-DDoS features available in AWS and how these features can be used jointly to help protect your services and applications.
DDoS has the potential to impact that availability of services and applications. If you put no defense in place, even a small DDoS can have a significant impact on that availability. We are going to look at techniques on AWS that you can use to deploy a DDoS resilient service.

AWS Shield

- **AWS Shield** features:
  - Is a managed distributed denial of service (DDoS) protection service
  - Safeguards applications running on AWS
  - Provides always-on detection and automatic inline mitigations
  - *AWS Shield Standard* enabled for at no additional cost. *AWS Shield Advanced* is an optional paid service.
- Use it to **minimize application downtime and latency.**

AWS Shield

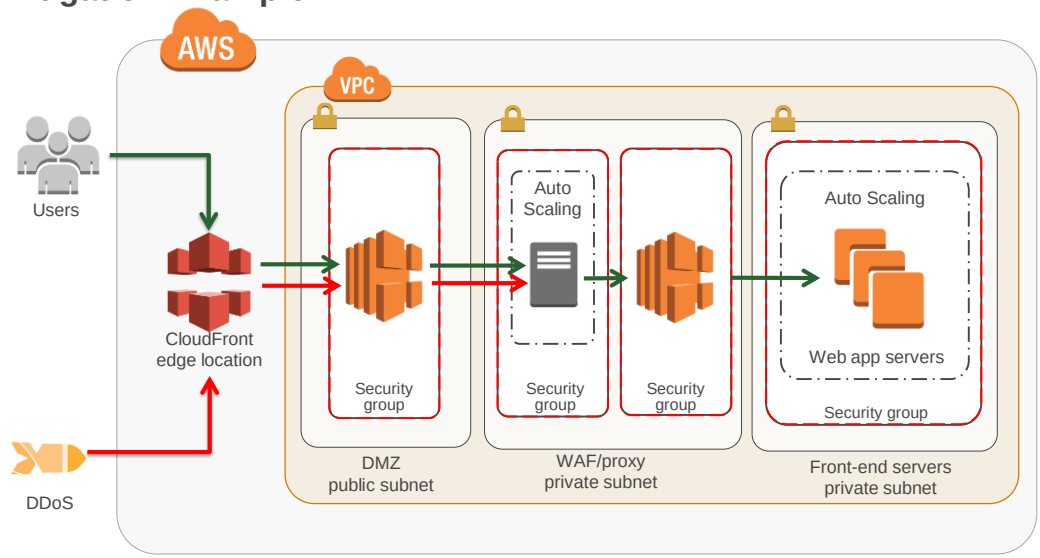
AWS Shield is a managed distributed denial of service (DDoS) protection service that safeguards applications that run on AWS. It provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection.

AWS Shield helps protects your website from all types of DDoS attacks, including Infrastructure layer attacks (like User Datagram Protocol—or UDP—floods), state exhaustion attacks (like TCP SYN floods), and application-layer attacks (like HTTP GET or POST floods). For examples, see the AWS WAF and AWS Shield Advanced Developer Guide.

AWS Shield Standard is automatically enabled to all AWS customers at no additional cost.

AWS Shield Advanced is an optional paid service. AWS Shield Advanced provides additional protections against more sophisticated and larger attacks for your applications that run on Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator, and Amazon Route 53. AWS Shield Advanced is available to all customers. However, to contact the DDoS Response Team, customers need to have either Enterprise Support or Business Support from AWS Support.

DDoS Mitigation Example



This is an example of a resilient architecture that can help to prevent or mitigate DDoS attacks.

The strategy to minimize the attack surface area is to (a) reduce the number of necessary Internet entry points, (b) eliminate non-critical Internet entry points, (c) separate end user traffic from management traffic, (d) obfuscate necessary Internet entry points to the level that untrusted end users cannot access them, and (e) decouple Internet entry points to minimize the effects of attacks. This strategy can be accomplished with Amazon Virtual Private Cloud.

Within AWS, you can take advantage of two forms of scaling: horizontal and vertical scaling. In terms of DDoS, there are three ways to take advantage of scaling in AWS: (1) select the appropriate instance types for your application, (2) configure services such as Elastic Load Balancing and Auto Scaling to automatically scale, and (3) use the inherent scale built into the AWS global services like Amazon CloudFront and Amazon Route 53.

Because ELB only supports valid TCP requests, DDoS attacks such as UDP and SYN floods are not able to reach your instances.

You can set a condition to incrementally add new instances to the Auto Scaling group when network traffic is high (typical of DDoS attacks).

Amazon CloudFront also has filtering capabilities to ensure that only valid TCP connections and HTTP requests are made while dropping invalid requests.

A WAF (web application firewall) is a tool that applies a set of rules to HTTP traffic, in order

to filter web requests based on data such as IP addresses, HTTP headers, HTTP body, or URI strings. They can be useful for mitigating DDoS attacks by offloading illegitimate traffic.
AWS now offers a managed WAF service. For more on AWS WAF, see:
http://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html
Whitepaper: AWS Best Practices for DDoS Resiliency:
https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- **Securing the System**
  - DDOS /AWS Shield , **AWS WAF**
- Auditing
  - AWS CloudTrail, AWS Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

AWS WAF (Web Application Firewall)

- AWS WAF is a web application firewall that lets you monitor web requests that are forwarded to Amazon CloudFront distributions or an Application Load Balancer.
- You can also use AWS WAF to block or allow requests based on conditions that you specify, such as the IP addresses that requests originate from or values in the requests.

SWINBURNE UNIVERSITY OF TECHNOLOGY

This week - Security

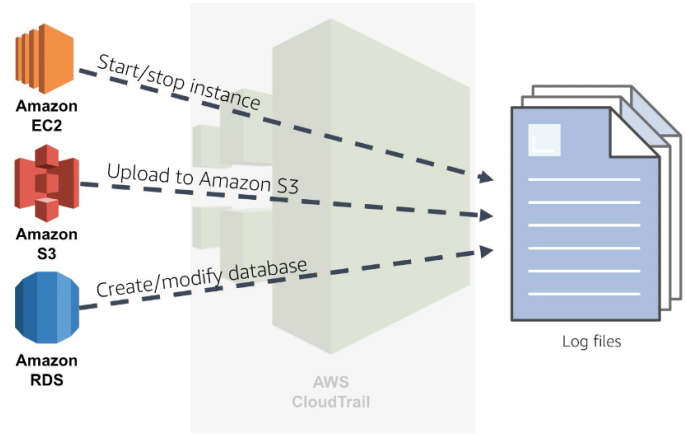
- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the Application
  - DDOS, AWS WAF
- **Auditing**
  - **AWS CloudTrail**, AWS Config

SWINBURNE UNIVERSITY OF TECHNOLOGY

82

Introduction to CloudTrail

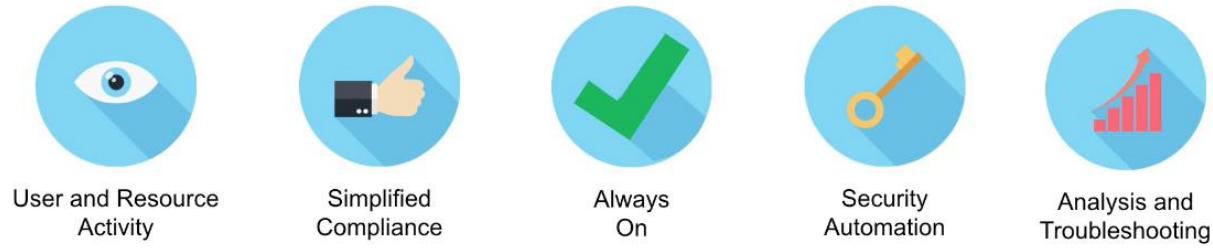
CloudTrail is a web service that records API calls for your account and delivers log files to you.



CloudTrail helps you log the API calls made in your AWS account across regions. Because everything in AWS is an API call, activity to AWS resources like starting and stopping instances, creating or modifying Amazon RDS databases, or uploading a file to Amazon S3 are logged, whether that action was performed via the CLI, an SDK, the console, or an API directly. This is a crucial tool for simplifying your governance, compliance, and risk auditing.

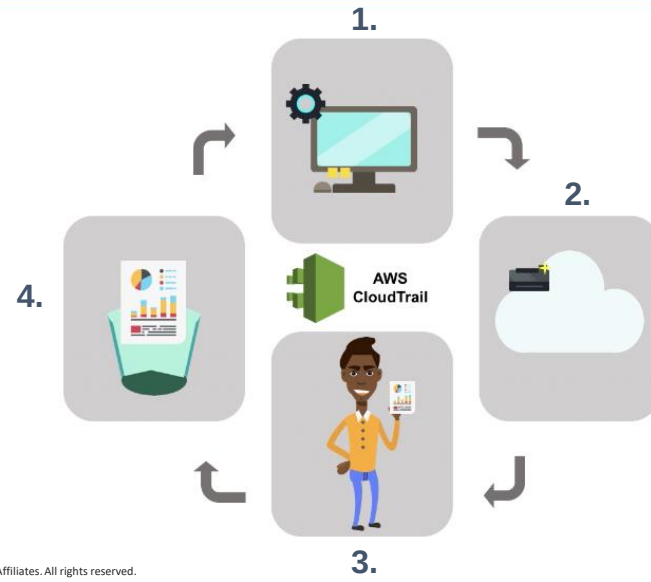
CloudTrail enables you to simplify governance, compliance, and risk auditing. The service accelerates analysis of operational and security issues by providing visibility into both API and non-API actions in your AWS account. With CloudWatch Logs integration, support for multi-region configurations, and log file integrity validation, CloudTrail provides comprehensive, secure, and searchable event history of activity made with the console, AWS SDKs, command line tools, and other AWS services.

AWS CloudTrail Benefits



CloudTrail has several key benefits. It increases your visibility into user and resource activity, which allows you to identify who did what and when in your account. Compliance audits are simplified because they are automatically recording and storing event logs. This allows you to search through log data, identify actions that are out of compliance, accelerate investigations into incidents and then expedite response. Because you are able to capture a comprehensive history of changes made within your account, you can discover and troubleshoot any operational issues in your account.

AWS CloudTrail Overview



How does this work?

First, an activity happens in your account. Next, CloudTrail captures and records that activity and calls it a *CloudTrail event*. The event will contain details about who performed the request, the date and time of the request, the source IP and how the request was made, the action being performed, the region in which the action was taken, and the response. By default, the logs are stored for 7 days. The activity log can be sent to other AWS services, so the activity history can be retained for as long as you like.

Using CloudTrail Best Practices

- Turn on CloudTrail log file validation
- Aggregate log files to a single Amazon S3 bucket
- Ensure that it is enabled across AWS globally
- Restrict access to CloudTrail Amazon S3 buckets
- Integrate with Amazon CloudWatch

To get the most out of CloudTrail, turn on CloudTrail log file validations. When you are configuring CloudTrail, you can aggregate all log files to a single Amazon S3 bucket. Additionally, a configuration that applies to all regions ensures that your settings are applied consistently across all existing and newly launched regions. You can also validate the integrity of log files by detecting whether or not they were changed or deleted after they were sent to the S3 bucket. It is also a good idea to run MFA to delete a CloudTrail bucket. This can be accomplished by restricting access to where they are stored. Lastly, integrating this service with Amazon CloudWatch enables you to define actions to execute when specific events are logged by CloudTrail.

This week - Security

- Access Control concepts
- Security in the Cloud: AWS Shared Responsibility Model
- AWS IAM
  - User Authentication, Groups, Roles
  - Authorization using Policies
  - Securing your AWS account
- AWS Authentication
  - Cognito, Directory service, STS, Web Identity
- Securing Data
  - In transit, at rest.
  - Encryption
- Securing the System
  - DDOS, AWS WAF
- **Auditing**
  - AWS CloudTrail, **AWS Config**

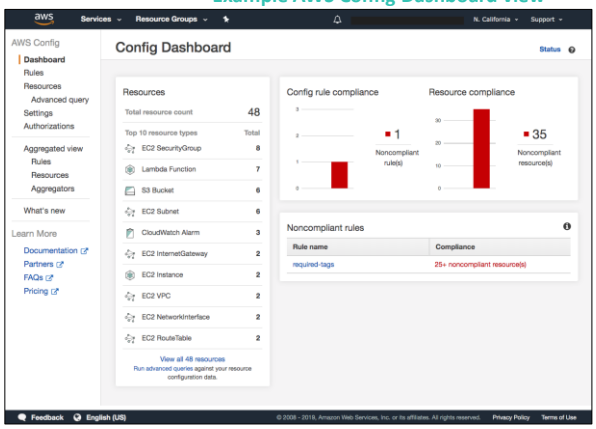
SWINBURNE UNIVERSITY OF TECHNOLOGY

87

AWS Config

AWS Config

Example AWS Config Dashboard view



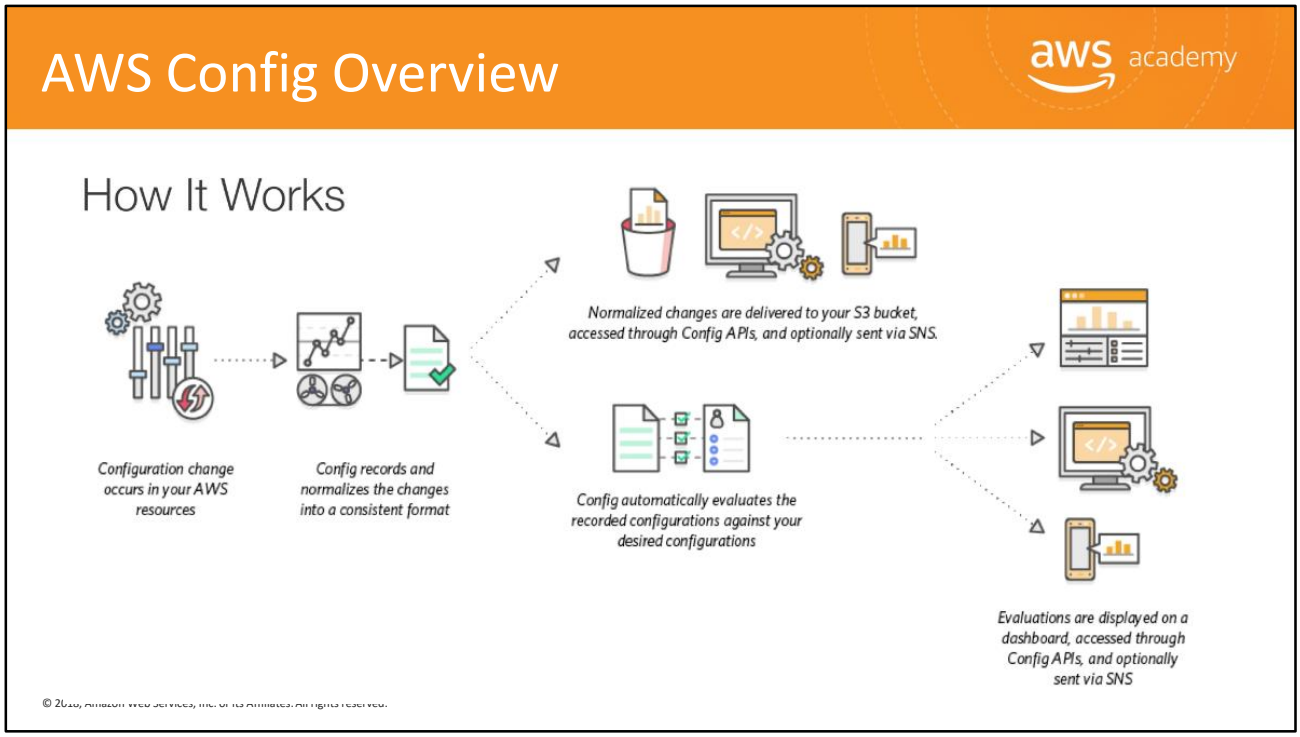
- **Assess, audit, and evaluate the configurations of AWS resources.**
- Use for continuous monitoring of configurations.
- Automatically evaluate *recorded* configurations versus *desired* configurations.
- Review configuration changes.
- View detailed configuration histories.
- **Simplify compliance auditing and security analysis.**

AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations, and it enables you to automate the evaluation of recorded configurations against desired configurations. With AWS Config, you can review changes in configurations and relationships between AWS resources, review detailed resource configuration histories, and determine your overall compliance against the configurations that are specified in your internal guidelines. This enables you to simplify compliance auditing, security analysis, change management, and operational troubleshooting.

As you can see in the AWS Config Dashboard screen capture shown here, AWS Config keeps an inventory listing of all resources that exist in the account, and it then checks for configuration rule compliance and resource compliance. Resources that are found to be noncompliant are flagged, which alerts you to the configuration issues that should be addressed within the account.

AWS Config is a Regional service. To track resources across Regions, enable it in every Region that you use. AWS Config offers an aggregator feature that can show an aggregated view of resources across multiple Regions and even multiple accounts.

AWS Config Overview



So, how does this work?

Looking at this diagram from left to right, the first thing that happens is a change occurs in one of your AWS resources. Next, the AWS Config engine records and normalizes that change in a consistent format. Then those changes are delivered to an S3 bucket, they are assessed through the AWS Config APIs and, optionally, they can be sent out via a notification service like Amazon SMS.

AWS Config will automatically evaluate the recorded configuration against your desired configuration. Those evaluations will be displayed on the dashboard or they are accessible via the AWS Config APIs. They can also be sent out via Amazon SMS.