# Architecture 101:

## Encryption

Encryption is the process of taking plane text and converting to cypher text and converting cypher text into plain text. Plain text and cypher text can be text image or any other data.

Encryption is generally used an algorithm and one or more keys, it’s generally used to encrypt data at **rest** (when local files/data are encrypted)or **in** **transit**.

### Symmetrical

Here the same key is used for both encryption and decryption, hence transferring the key is the challenge here.

### Asymmetrical

Here there will be two keys, **public key** used for encryption and **private key** used for decryption. Vendor can generate and send the public key over anything as it doesn’t make any sense standalone. The party will always hold the private key to decrypt the message.

So private key will never become public sharing.

## RPO vs RTO

These are the parts of disaster recovery

### RPO

**Recovery Point Objective** where it decides how much business it can tolerate to lose in terms of time. This is the maximum time between a failure and last successful backup

### RTO

**Recovery Time Objective** is the amount of time a system can be down. How long a solution takes to recover

# AWS Architecture 101:

## AWS Global Infrastructure

### Region and Availability Zone

AWS have **Region** and **Availability Zone**, Data stored in a AWS region will not leave the region boundary because the data handling laws/policy could vary from region to region.

Region is identified by region-code and region-name.

Region contains multiple Availability Zone (AZ) are the data centers which are separated and isolated networks. A failure in on AZ won’t impact other which makes region fault tolerant.

AZs in same region are connected with redundant, high speed, low latency network connections.

Most AWS service runs within AZs. Some series operates from one AZ, while other replicate between AZs. Some service allows us to choose the AZ where some doesn’t.

### Edge Locations

Edge locations/Points of presence are small pockets of AWS compute, storage and networking close to major populations and generally used for edge computing and content delivery.

## AWS Well-Architected Framework

This is a set of best practice guidelines principles to architect AWS structure for our need. This framework provides some principles to follow.

**Security, Reliability, Performance Efficiency, Operational Excellence, Cost Optimization.**

Refer related white paper:

<https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf>

## Elasticity

With vertical scaling based on fluctuation of demand our capacity may not meet most of the time server either could be over burden or underutilized and with huge capacity resource wastage on non-peek time not avoidable.

With horizontal scaling, instead of a high capacity server we will have multiple low capacity server chained, but here too the demand to capacity need of underutilization and over burden can’t be avoided.

Where comes **Elasticity** which is an automated version of horizontal scaling where based on the demand hour capacity will be increased or decreased. Elasticity provides flexibility to scale in or scale out to very maximum to complete minimum.

Elasticity helps to achieve maximum cost optimization and performance efficiency.

# AWS Product Fundamentals

## S3 Simple Storage Service

S3 is a **global** object storage platform that can be used to store objects in form of text files, photos, audios, videos, large binaries and other object type.

The region for s3 is Global region. The basic unit of S3 is **bucket.** The bucket name should be unique globally.

Anything we will store inside S3 bucket will be replicated across the AZs in the region. Bucket is a flat store, so even we create a folder inside bucket still it’s an object got mapped.

Object (key value, key is name value is the data) will have a unique name inside an object. Object can have size till 5TB.

S3 is great for storage and also for internet access of our data stored.

S3 can’t be mounted to server as network drive or mount point. We shouldn’t attach S3 buckets to server as disk or block storage.

### S3 quick notes

* Bucket name should be globally unique.
* Minimum of 3 maximum of 63 chars – no uppercase or underscore.
* Must start with a lowercase or number but can’t formatted as an IP address.
* Default 100 buckets per account and hard 1,000 buckets limit via support request.
* Unlimited object and capacity inside a bucket.
* The objects key is a name and value is the data.
* An object size from 0 to 5TB.

## Introduction to CloudFormation

AWS CloudFormation is an automation product available in AWS as Infrastructure as Code (**IoC**) product. We can create, manage and remove infrastructure using template JSON or YAML.

So we can create a template to automate multiple redundant tasks of infrastructure create and management. Ex for Dev, QA so on.

### The three building blocks of CloudFormation

**Template**

Is a JSON or YAML contains logic resources and configurations

**Stack**

Stacks are created and modified based on template, which also can be used to change and update a stack.

**Physical Resources**

Stack takes logical resources from template and create, update and delete physical resources in AWS.

### Quick tips

* A template can create up to 200 resources.
* Change in logical resource update or replace the physical resources with some disruption.
* New logical resource cause new physical resource.

### Useful resources

Github file:

<https://github.com/linuxacademy/content-aws-csa2019/tree/master/lesson_files/01_aws_sa_fundamentals/introduction_to_cloudformation>

CloudFormation template:

<https://s3.amazonaws.com/cloudformation-templates-us-east-1/WordPress_Single_Instance.template>

AWS CloudFormation resource reference:

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

# Identity and Access Management (IAM)

## IAM Essentials

IAM is the primary service that handles authentication and authorization within AWS environment i.e. AWS api.

IAM controls access to AWS service via **policies** that can be attached to **users, groups** and **roles.** Users are given long term credentials to access AWS resources.

Roles allow for short-term access to resources when assumed, using temporary access credentials.

### ARN – Amazon Resource Name

ARN is a notation in AWS which can unique identify any resource under any user any region.

**Format of ARN:**  arn:partition:service:region:account-id:

Ex: arn:aws:s3:::myamazingcatpics/truffs.jpeg

arn:aws:ec2:us-east-1:1246863355:instance/\*

**Note:** Fields with :: omits the value and \* is a wildcard match

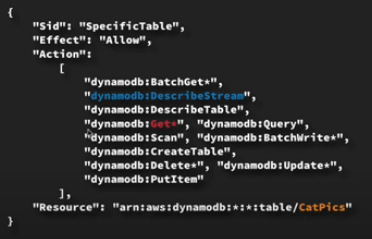
## IAM Policies

An IAM Policy (Policy Document) is known as an **identity policy** when attached to an identity or a **resource policy** when attached to a resource. They have no effect until they attached to something.

A policy document is a list of statements:



A statement matches a request to AWS. Requests are matched based on their **Action** which are API calls. A given statement results in an **Allow** or **Deny** for the request.



**Exam tips: IAM policies**

* If a request isn’t explicitly allowed, it’s implicitly (default) denied.
* If a request is explicitly denied, it overrides everything else.
* If a request is explicitly allowed, it’s allowed unless it’s explicitly denied
* Remember **DENY -> ALLOW -> DENY**
* Customer-managed policies are flexible but required administration.
* Inline and managed policies can apply to users, groups and roles.

## IAM Users

IAM users are type of IAM identity suitable for long-term access for a known entity (human, service, application).

Principals authenticate to IAM user either with a user name and password or access keys.

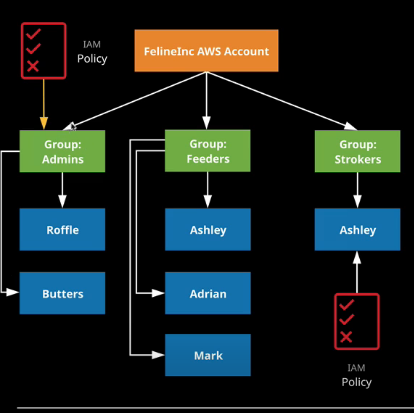
**Exam tips:**

* Hard limit 5,000 IAM users per account.
* 10 group membership per IAM user.
* Default maximum 10 policies per user.
* No inline limit, but you can’t exceed 2048 characters for all inline policies on a IAM user.
* 1 MFA per user.
* 2 access keys per user.

## IAM Groups

IAM Group is a collection of IAM users. Groups allows easier administration over sets of IAM users. Inline and managed policies can be applied to groups that flow on to members of that group.

Groups are not true identity, they can’t be a principal in a policy, so they can’t be used in resource policies.



**Exam Tips:**

* Groups are admin feature to group IAM users.
* Groups can contain many IAM users and users can be in many groups.
* IAM inline policies can be added to IAM groups and these flows into IAM users under that group.
* Managed IAM policies can be attached and flow onto IAM users who are members.
* Group is not the true identity and can’t be referenced from resource policies.
* Groups have no credential.

## IAM Access Keys

Access keys are pair of values used by applications, SDKs or the AWS command line to authenticate to AWS.

Access key contains two parts **access key ID** and **secret access key.** The access key ID is the public part of the key and is stored by AWS once generated.

The secret access key is private part should be stored by user and never be shared and neither can be changed if lost.

An IAM user is the only identity that uses access keys. They are allowed two sets. They can be created, deleted, enabled and disabled.

Access keys can’t be used to login to console, and they don’t expire. If anyone finds a set of access keys, they have access to the permission of the IAM users to which they belongs.

An user can only have 2 access keys.

## IAM Roles

IAM roles are **assumed** by another identity allowed in the **trust policy** – IAM user, AWS service, another AWS account, web identity or even anonymous identity. When a role assumed the Security Token Service (**STS**) generates a **time-limited** set of access keys (temporary security credentials). These access keys have permission defined in permissions policy. IAM roles have no long term credentials (access keys or user name or password).

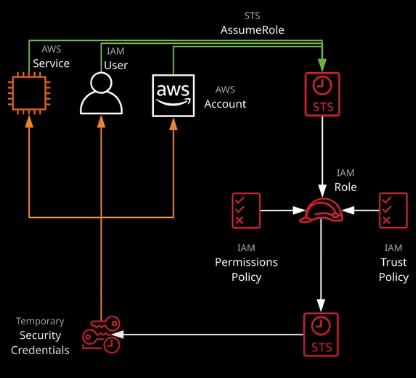
**Trust policy** controls which identities can assume the role.

**Permissions** policy defines the permissions provided.

Example Scenario:

* Company merger
* AWS service access
* “Break glass” style extra access
* Cross-account access
* Web identity federation

Architecture (How IAM role works)



# Server-Based Compute (EC2) Fundamentals

## EC2 Architecture

