==========

AWS Introduction

==========

-> AWS stands for Amazon Webservices

-> AWS cloud managing by Amazon company

-> AWS is one of the leading Cloud Provider in the market

-> AWS started providing IT resources over internet from 2006 onwards

-> 190+ countries using AWS Cloud

-> AWS providing 200+ Services

-> AWS providing Cloud Services based on 'Pay As You Go' model

==========

AWS Services Names

==========

EC2 : Elastic Compute Cloud : To create virtual machines

EBS : Elastic Block Store (External HD)

EFS : Elastic File System

S3 : Simple Storage Service : Unlimited Storage

RDS : Relational Database System : To create SQL Databases (Oracle, MySQL, Postgres, MS SQL etc..)

VPC : Virtual Private Cloud : Isolated Network

Route 53 : Domain Name Mapping (URL Mapping)

BeanStalk : For Paas Model

IAM : Identity & Access Management (who can access which service in AWS)

ECS : Elastic Container service (To run containers)

ELB : Elastic Load Balancer (Load Balancing)

Lambda : Serverless Computing (run the code without thinking about servers)

++

-> To use AWS provided cloud services we need to create one account in AWS

Note: It will ask debit / credit card for account creation

-> AWS will charge 2 rs for account creation and they will send 2 rs back to our account after account verified

-> In AWS few services are free and few services are paid

-> As part of our training we will use both free and paid services

Note: When we use paid services, after practise completion we need to delete those service to avoid billing

-> If bill got generated we can request AWS Support team to waveoff our bill because we are AWS learners and we are exploring AWS Cloud services.

-> AWS will not deduct bill amount from our card directley. We need to pay that bill manually.

-> If we don't pay AWS bill amount then our AWS account will be terminated.

++

-> AWS providing global infrastructure

-> 190+ contries are using AWS Cloud through internet

-> To provide Global Infrastructure AWS using Regions & Availability Zones concept

-> Region means one geograhical location

-> Availability Zone means data center

-> Data Center means a big building which contains servers with network

-> One Region can have multiple Availability Zones (AZ)

-> AWS Having 26 Regions and 84 Availability Zones in the world

-> In india AWS having 1 region (Mumbai) ---- ap-south-1

-> Mumbai region having 3 availability zones

- ap-south-1a
- ap-south-1b
- ap-south-1c

Note: Hyd Region Coming Soon

Note: It is recommended to use Nearest Region in AWS to setup our infrastructure

Note: In AWS few services are global (S3, Route 53 etc...) and few services regional (EC2, VPC, RDS etc...)

========

EC2

========

-> EC2 stands for " Elastic Compute Cloud "

-> EC2 is the most demanded service in AWS

-> EC2 is used to create virtual machines

-> EC2 instances are re-sizable

-> EC2 is regional service

-> EC2 instance minimum billing period is 1 hour

Ex: If you run for 5 minutes also it will be considered as 1 hour

-> The Virtual Machine which we create using EC2 service is called as EC2 instance

Alias Names : EC2 Instance / Virtual Machine / VM / Server

-> EC2 is a paid service

Note: EC2 providing t2.micro as free tier eligible for 1 year with monthly 750 hours

-> When we create EC2 instance, AWS will provide default storage and default network

-> Storage will be provided by EBS (Elastic Block Store)

EBS Max Size : 16 TB (16000 GB)

Linux : 8 GB (free tier, default)

Windows : 30 GB (free tier, default)

-> Network will be provided by VPC (Virtual Private Cloud)

==========

EC2 instances types

==========

1) On-Demanded Instances

2) Reserved Instances

3) Spot Instances

4) Dedicated Host

==========

OnDemanded Instance

==========

-> Whenever we want then we can create it

-> Fixed Price (Hourly)

-> Pay For Use

-> No Prior Payments

-> No Committment

==========

Reserved Instances

==========

-> It is like advanced booking

-> Long Term Commitment

-> Prior Payment option available (paritial payment / full payment)

-> Commitment for 1 year or 3 years

-> AWS will provide discount on price

==========

Spot Instances

==========

-> It is like bidding or Auction

-> AWS will offer high capacity systems for low price

-> 72% savings on price

==========

Dedicated Host Instance

==========

-> This is a physical machine

-> Licensed Softwares

-> It is very costly when compared with other instance types

==========

EC2 instance states

==========

Running --- Billing

Stopped -- No Bill

Terminated -- No Bill

==========

EC2 instance families

==========

t2. t3, t4, c2, c3, i2, i3, m2, m3, m4 etc....

-> We are using t2.micro instance for practise

Note : t2.micro instances are Free for 1 year (from aws account creation date)

Note: Monthly 750 hours usage is free for 1 year when we go for t2.micro

Note: We can find EC2 instance types service rates in dashboard

==========

Amazon Machine Image (AMI)

==========

-> AMI is a template to launch our instances

-> Image represents pre-configured system with softwares installed

-> To create EC2 instance with Windows OS we can use Windows OS AMI

-> To create EC2 instance with Linux OS we can use Linux OS AMI

-> In AWS, we can create our own AMI also

==========

What is Key Pair in EC2 ?

==========

-> Key Pair is the combination of Public Key and Private key

-> AWS will store public key and it will provide private key for us (We have to save that very carefully)

-> Private Key file extension will be .pem

-> Public key & Private Key is used to connect with EC2 instance securley

-> If we want to connect with EC2 VM we need to provide private key for AWS then AWS will compare our private key with its public key. If both keys are matched then only AWS will allow to login into EC2 VM.

Note: One key pair we can use to create multiple instances

Note: Key-Pairs are free of cost (No Bill)

Note: Key pair we can use for any OS Based VM

-> Key Pairs are regional specific

-> Can i change key pair after EC2 VM creation ------> Not possible

==========

Security Groups

==========

-> Security Group is like a virtual firewall for our EC2 instance

-> Security Group will control Incoming and Outgoing traffic of our EC2 instances

-> To allow the incoming traffic we will configure Inbound Rules

-> To allow the outgoing traffic we will configure Outbound Rules

RDP : 3389

SSH : 22

HTTP : 80

HTTPS : 443

NFS : 2049

Note: Security Groups are free of cost

Note: One security group we can use for Multiple Instances also

-> Security Groups are VPC specific

==========

Creating Windows Virtual Machine

==========

1) Goto EC2 service

2) Launch instance

3) Select AMI (Windows - Free tier eligible)

4) Select Instance Type (t2.micro - Free tier eligible)

5) Storage (Default 30 GB - Free tier)

6) Network (Default VPC)

7) Create New Pair

8) Create New security group with RDP protocol

=> Once EC2 VM created then click on 'Connect' button and get below details

DNS : ec2-15-207-89-254.ap-south-1.compute.amazonaws.com

Username : Administrator

Password : rcAF2=D3s%g&%O98o?)*xUYd&!vdw?dp

-> Connect to Windows VM using RDP

==========

Launching Linux Virtual Machine in AWS

==========

1) Goto EC2 service

2) Launch instance

3) Select AMI (Amazon Linux - Free tier eligible)

4) Select Instance Type (t2.micro - Free tier eligible)

5) Storage (Default 8 GB - Free tier)

6) Network (Default VPC)

7) Create / Use Key Pair

8) Create New security group with SSH protocol

=> Once EC2 VM created then click on 'Connect' button and get details

=> Connec to Linux VM using MobaXterm

==========

Instance States

==========

Running ---- Bill (hourly)

Stopped ------- No Bill

Terminate ------- No Bill

9: 00 AM (start) ---- 9:05 AM (stopped) ---------> 1 hr

9:10 AM (start) ----> 9:15 AM (stopped) -------> 1 hr

9:30 AM (start) -----> 9:45 AM (stopped) --------> 1 hr

==========

Types of IP's in AWS

==========

-> IP stands for Internet Protocol

-> Every Machine should have one IP address

-> IP is like an address for computer

-> AWS providing 3 types of IPs

1) private ip

2) public ip

3) elastic ip (public static ip)

-> When we launch EC2 instance then AWS will provide one private ip and one public ip for our instance

-> Private IP is a fixed IP and it is used by AWS for internal purpose. It will not change when we re-start our EC2 instance.

-> Public IP is a dynamic IP. When we re-start our EC2 instance new Public IP will be generated.

Note: To connect with EC2 instance from outside we will use Public IP.

-> Elastic IP means fixed public IP address.

-> We can create Elastic IP and we can associate that elastic ip for our EC2 instance

-> Elastic IP address will not change when we re-start our ec2 instance

Note: Elastic IPs are commercial (paid)

Working process

++++++++++++++++

1) Create Elastic IP

2) Associate Elastic IP for EC2 instance

3) If we don't want to use elastic ip then De-Associate Elastic IP from EC2 instance

4) Release Elastic IP to AWS (Its mandatory)

***** Note: If we create Elastic IP then we have to associate that Elastic IP otherwise bill will be generated for that******

(We shouldn't keep Elastic IP as un-used ip)

================

Load Balancing

==================

-> If we deploy our application in one server then burden will increase on that server

-> If burden increased on server then below are the problems we are going to face

1) Request processing will become slow

2) Responses will be delayed for customers

3) Server might get crash

4) Brand / Trust issues on our business

5) Revenue Loss

6) Single point of failure (if server is down then application will be down)

Note: Good Business needs Good website

-> To overcome all these problems we will run our application in Multiple Servers

-> The process of running our application in Multiple Servers is called as Load Balancing.

-> To implement Load Balacing we will use Elastic Load Balancer (ELB) in AWS

-> LBR will recieve the request and it will distribute the requests to servers in round robbin fashion

==========

Types of Load Balancers in AWS

==========

1) Application Load Balancer (ALB)

2) Network Load Balancer (NLB)

3) Gateway Load Balancer (GLB)

4) Classic Load Balancer (old, getting retried on 15-Aug-2022)

-> To implement load balancing for HTTP & HTTPS requests we will go for Application Load Balancer (ALB)

-> By using Application Load Balancer we can implement Path Based Routing

==========

Implemenging Load Balancer

==========

1) Create 1st EC2 intance with below user data script

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 1</h1></html>" > index.html
```

service httpd start

2) Create 2nd EC2 intance with below user data script

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 2</h1></html>" > index.html
service httpd start
```

3) Create 3rd EC2 intance with below user data script

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 3</h1></html>" > index.html
service httpd start
```

4) Create Target Group with above 3 EC2 instances

(Target Group means group of servers which are running our aplication)

5) Create Application Load Balancer using Target Group with 'internet-facing' scheme

6) Access the application Load Balancer DNS URL

Note: When request comes to Load Balancer it will distribute the requests to servers which are part of given target group.

==========

Monolith Vs Microservices

==========

-> Application can be in 2 ways

1) Monolith Architecture

2) Microservices Architecture

-> Monolith Architecture means all the functionalities will be developed in single project

-> A big war file will be created

-> Monolith Architecture based project is difficult to maintain

-> For any small change in the code then we have to re-deploy entire application

-> Single Point Of failure

Note: To overcome the problems of Monolith Architecture we are using Microservices Architecture

-> Microservices is an architectural design pattern

-> Micservices architecture means project functionalities will be developed in multiple apis

-> Every API is called as one project

-> Every API contains limited functionality

-> Making changes to the functionality is easy in microservices

-> Maintenence of the project will become easy

-> For any code changes we no need re-deploy all the apis (deploy only changed api)

Note: For Monolith app load balancing one target group will be created and application will be deployed in all the servers who are belong that target group.

==========

Microservices Based Load Balancing

==========

-> Multiple APIs will be available In Microservices Architecure

-> Every API is called as one microservice

-> For every microservice one target group will be created

Hotels API ===> Hotels Target Group with 3 servers

Flights API ===> Flights Target Group with 3 servers

Trains API ===> Trains Target Group with 3 servers

==========

How to implement LBR for Microservices based application

==========

1) Create EC2 Instance with below user-data (Name it as HotelServer-1)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotel Server - 1</h1></html>" > index.html
service httpd start
```

2) Create EC2 instance with below user-data (Name it as HotelServer-2)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotels Server - 2</h1></html>" > index.html
service httpd start
```

3) Create HotelServers Target group with above 2 instances

4) Create EC2 instance with below user-data (Name it as FlightServer-1)

```
#! /bin/bash
sudo su
yum install httpd -y
```

cd /var/www/html

echo "<html><h1>Flights Server - 1</h1></html>" > index.html

service httpd start

4) Create EC2 instance with below user-data (Name it as FlightServer-2)

#! /bin/bash

sudo su

yum install httpd -y

cd /var/www/html

echo "<html><h1>Flights Server - 2</h1></html>" > index.html

service httpd start

5) Create FlightsServers Target group with above 2 instances

6) Create Load Balancer by select HotelServers Target Group

7) Goto LBR Listeners and configure Route Based Routing for Flights Target Group

8) Test it with DNS name

Note: Once practise completed, delete LBR, Target Groups and EC2 instances

==========

Auto Scaling

==========

=> AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

=> Using AWS Auto Scaling, itâ€™s easy to setup application scaling for multiple resources across multiple services in minutes.

=> Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.

=> The process of increasing and decreasing no.of ec2 instances based on the load is called as Auto Scaling.

==========

Advantages with Auto Scaling

==========

1) Fault Tolerenece : Detect when an instance is unhealthy, terminate it, and launch an instance to replace it.

2) Availability : Ensure that your application always has the right amount of capacity to handle the current traffic demand.

3) Cost Management : Save money by dynamically launching instances when they are needed and terminate them when they aren't needed.

==========

How to setup Auto Scaling Group

==========

1) Create Launch Template

2) Create AutoScaling Group with Launch Template

3) Configure Desired, Min and Max Capacity

4) Attach AutoScaling Group to particular Target Group

5) Configure Scaling Policy

################################

Simple Storage Service (S3)

################################

######### Amazon S3 : Object storage built to store & retrieve any amount of data from anywhere ########

######### Note: 5 GB of S3 standard storage for 12 months with the AWS Free Tier ########

-> S3 is a storage service in AWS cloud

-> S3 is unlimited storage

-> S3 is object based storage

-> In s3 we can store all flat files (any type of file)

-> We can upload, download and access files from S3 at any point of time

-> The files in s3 can't be executed

-> We can't install OS, DB etc in S3

-> We can't attach S3 to EC2 instance but we can access s3 buckets data from EC2 instance

-> S3 supports static website hosting

-> S3 is cheaper than EC2

-> S3 is serverless

-> In S3 we will store data in buckets. Bucket is a container & bucket contains objects

object = file

key is name of the object

-> S3 is global but buckets are regional

-> Bucket names are universal or unique

Note: always create a bucket with your company name or project name.

-> We can't create one bucket inside another bucket

-> We can create multiple buckets in multiple regions

-> Max no.of buckets you can create in S3 is 100 (soft limit)

-> By default buckets are private, if required we can make it public.

create bucket -> inside that create folder called photos -> inside that upload puppy.jpg

Note: Every object will have its own url/endpoint.

Ex: http://8pmbukcet.s3.amazonaws.com/photos/puppy.jpg

bucketname+ domain + object name

-> S3 uses WORM model (Write Once and Read Many)

==========

Static Website Hosting using S3

==========

=> Collection of web pages is called as Website

=> Websites are 2 types

1) Static Website

2) Dynamic Website

=> The website which will display same content/response for every user is called as Static website.

=> The website which will display response based on user request is called as Dynamic Website.

########## Website Code Git Repo : https://github.com/ashokitschool/s3_static_website_hosting.git ###########

1) Create a bucket in S3

- Enter unique name for bucket

- uncheck block public access

2) Upload Website content files in bucket (assets folder, index.html and error.html) make sure you given public access for files which we are uploading.

3) Go to Bucket Properties tab -> Enable Static Website hosting and configure index and error pages

index.html for main content

error.html for wrong url

4) It will display URL, access that URL

##############

Versioning

##############

-> Versioning is like a backup tool in S3

-> By default versioning is not enabled on the bucket

-> Versioning is enabled on bucket level but applied on object level.

-> When we upload same file multiple times then multiple versions will be created.

corejava.jpg (v1, v2, v3) - v3 latest version

advjava.jpg (v1, v2, v3) - v3 is latest version

-> Version ID is always unqiue

-> versioning files we can download at any time

-> AWS charges for Versioning, becareful while you enable versioning for huge files.

-> S3 is unlimited storage

Min obj size = 0 Bytes

Max obj size = 5 TB

-> We can have unlimited no.of 5TB objs in a single bucket

###############

Storage Classes

###############

-> While uploading the objects into S3 , selecting storage class is mandatory

Scenario : some customers wants to store and wants to access data frequently... some other customers wants to store data but don't want to access frequently... we can't charge same bill for both customers because they have diff businss requirements.

-> To meet business requirements of clients, AWS provided several storage classes in S3

Standard Frequently Access(FA)

This is used for frequently access data

Default storage class

Regular purpose (storing, website, images etc)

No retrival charges

Availability = 99.9 %

Durability=999999999 (11 9's)

Min Obj size is 0 Bytes

Standard In-Frequently Access (IA)

Frequently access but not critical

Not Retrival charges

AWS doesn't recommend to use this

Cheaper than others

Availability = 99.9 %

Durability=99.99%

min obj size = 128 kb

min duration : 30 days

One Zone IA

In-frequently access but not critical

retrival charges apply

availability=99.99%

Durability= 11 9's

Min Obj size = 128 KB

min obj size = 128 kb

min duration : 30 days

Intelligent Tiering

Unknown access pattern

Based on access it moves from FA to IA

availability=99.99%

Durability= 11 9's

min duration : 30 days

Glacier

Infrequently access data

archiving purpose

vault : container of archives

Archive : Object /Archive(zip) -> 40 TB

unlimited no.of archives

1000 vaults

Retrival charges apply

Glacier has retrival options

Expedited : 1 to 5 mins

Standard : 3 to 5 hours

Bulk : 5 to 12 hours

availability=99.99%

Durability= 11 9's

Min duration : 90 days

Note: Deep Glocier min duration is 180 days

-> It is possible to move the objects from one storage class to another storage class automatically (LCM)
-> Life Cycle management.

-> LCM is created on bucket level and applied on object level

-> Life cycle rule (current version & previous version)

-> my obj moving from FA -> IA (30 days) -> Glacier(60 days) ->this is called transition

-> Delete after 365 days (Expiration)

-> Object Lock (Permanent Lock & certain period lock)

-> I have a bucket named as movies.

-> We can enable bucket logs to identify who is accessing our bucket

---

Encryption

Encryption is used for security

Encryption can be done in 2 ways

In-Transit : Encryption while data is moving/flow HTTPS

Data At Rest : Encryption while data is at rest

-> Amazon S3 has 3 types of encryptions

serer-side encryption

SSE - S3 (AWS Managed Key)

SSE - KMS (AWS KMS Key)

SSE - C (Customer Provided Key)

client-side encryption (should be handled by customer, how to reach aws is our headache)

in-transit encryption (using https)

-> AWS Certification manager (ACM)

is where you can generate HTTPS certificates (SSL/TLS/HTTPS)

-> S3 data consistency models - 2 types

Read after write consistency for PUTS of new objects (immediatley)

Eventually consistency for overwrites of puts and deletes

-> Pre-Signed URL (it will be accessible for limited time)

##########

Transfer Acceleration

##########

-> If we want to transfer the data from our place to AWS S3 bucket it will use our own internet.

-> We can speed up transfer process using Transfer Acceleration.

-> It is used to transfer data fastly (It will use CDN concept) With CDN it will use AWS internal network.

##################################

RDS (Relational Database Service)

##################################

-> Every application will contain 3 layers

1) Front End (User interface)

2) Back End (Business Logic)

3) Database

-> End users will communicate with our application using frontend (user interface).

-> When end-user perform some operation in the front-end then it will send request to backend. Backend contains business logic of the application.

-> Backend logic will communicate with Database to perform DB operations.

(insert / update / retrieve / delete)

##

Challenges with Database Setup on our own

##

1) Setup Machine to install Database Server

2) Purchase Database Server License

3) Install Database Server in our Machine

4) Setup Network for our machine

5) Setup power for machine

6) Setup a server room to keep our machines

7) Setup AC for room for cool temperature

8) Setup Security for room

9) Setup Database backups

10) Disaster Recovery

-> If we use AWS cloud, then AWS will take care of all the above works which are required to setup Database for our application.

-> In AWS, we have RDS service to create and setup database required for our applications.

-> We just need to pay the money and use Database using AWS RDS service. DB setup and maintenence will be taken care by AWS people.

RDS is full managed by AWS

** Using RDS we can Easily set up, operate, and scale a relational database in the cloud ***

##

Steps to create MYSQL DB using AWS RDS

##

1) Login into AWS management console

2) Goto RDS Service

3) Click on 'Create Database'

Choose a database creation method : Easy Create

Engine Option : MySQL

Template : Free Tier

DB instance Identifier : ihis (Note : you can give anything)

Username : admin

Password : Choose a passord

4) Click on 'Create Database' (It will take few minutes of time to create)

Note: Notedown username and password of the database

5) Once Database created, it will provide database Endpoint URL to access

6) Change Database to Public Access

7) Enable All Traffic in Security Group attached to Database.

####################

MySQL DB Properties

####################

Endpoint : database-1.cbair9bd9y7d.ap-south-1.rds.amazonaws.com

Uname : admin

Pwd : ashokitdevdb

Port : 3306 (it is default port for mysql db)

Note: We need to provide DB properties to project development team / testing team

##############################

Steps to test MYSQL DB Setup

##############################

1) Download and install Visual Studio using below link

Link : https://aka.ms/vs/17/release/vc_redist.x64.exe

2) Download and install MySQL Workbench using below link

Link : https://dev.mysql.com/downloads/workbench/

3) Create Database Connection in MySQL workbench using Database properties

4) Once we are able to connect with Database then we can execute below queries in Workbench

###############

MySQL Queries

###############

```
show databases;

create database sbidb;

use sbidb;
```

show tables;

create table emp_dtls(emp_id integer(10), emp_name varchar(20), emp_salary integer(10));

insert into emp_dtls values(101, 'Raju', 5000);

insert into emp_dtls values(102, 'Rani', 6000);

insert into emp_dtls values(103, 'Ashok', 7000);

select * from emp_dtls;

####################################

Working with MySQL client in Ubuntu

####################################

$ sudo apt-get update

$ sudo apt-get install mysql-client

$ mysql -h <endpoint> -u <username> -p (click enter and give password)

##

Working with MySQL client in AMAZON Linux

##

$ sudo yum update

$ sudo yum install mysql

$ mysql -h <endpoint-url> -u <username> -p (click enter and give password)

-h : It represents host (DB edpoint URL)

-u : It represents DB username

-p : It represents DB password

Ex: mysql -h database-1.cbair9bd9y7d.ap-south-1.rds.amazonaws.com -u admin -p

=> We can execute below queries to see the data which we have inserted previously using Workbench.

```
show databases;

use sbidb;

show tables;

select * from emp_dtls;
```

##################################

Spring Boot App with AWS RDS DB

##################################

URL : https://youtu.be/GSu1g9jvFhY

==========

IAM (Identity & Access Management)

==========

-> An AWS Identity and Access Management (IAM) user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS Services.

-> AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.

We can use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

-> IAM helps protect against security breaches by allowing administrators to automate numerous user account related tasks.

-> Best practice of using the root user only to create your first IAM user.

-> Enable Multi Factory Authentication (MFA) for Root Usr

-> By using Google Authenticator App we can configure "Virtual MFA"

==========

Best Practices:

==========

- When we login AWS using 'email' and 'password', that has complete access to all AWS services and resources in the account (Root account).

- Strongly recommended that you do not use the "root user" for your everyday tasks, even the administrative ones.

- Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

- IAM user is truely global, i.e, once IAM user is created it can be accessible in all the regions in AWS.

- Amazon S3 also considered as Global but, it is not truely global. When we create a bucket in S3

it displays all the buckets of other regions in one place , so that is the reason we are calling AmazonS3 is Global (but partly global).

- But IAM is 100% Global. Once you create IAM user you can use it anywhere in all the regions.

1. Main things in IAM is

    -Roles

    -Users

    -Policies / Permissions

    -Groups

2. IAM users can be accessible by the following 3 ways.

    -through AWS console

    -CLI (Command Line Interface)

    -API (fast glaciers)

3. In MNCs , permissions will not be provided for individual users. Create the Groups and add the users into it.

Users & Groups are for the Endusers.

Roles are for the AWS Services.

Steps:

====

1. Create an IAM user

Services - Security, Identity, & Compliance - IAM

Users---<Add user>

User name* = Iamuser1

Access type = 'select' both "Programmatic Access"

"AWS Management Console access"

Console password = 'select'

custom Password = (********somepassword eg:test1234)

click <NextPermissions>

(Note: we are not providing any permissions as of now, just <create user>)

Once the IAM user has been created.

AccessKeyID =AKIAIEJH7Z3FDKH36YWQ

Secretaccesskey=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

(Note: Once you close this window, AccessKeyID and Secret Accesskey has gone, so save it somewhere)

- Best Practice is never give an individual permissions to the user, as users will be changed frequently, when they left the organization.

So Need to create the Groups and assign the users to it.

2. Group

<create new group>

Groupname =admins

(Note: no need add any policy now).

<creategroup>

3. Add user to this group

click on newly create group 'admins'

<Add users to Group>

GroupARN =arn:aws:iam::540105522204:group/admins

-Always add the permissions to the 'Groups' level not to the 'users' level. Its a Best Practice in the real-time.

****

Policies:

****

- When we want to add the permissions to the the groups is through the 'Policies'.
- Default AWS Policies are appear in'Orange color Icons'
- One disadvantage of AWS Default Policies are , we can't customize the policies to apply to the Groups.
- To provide customized policies to apply to Groups, we need to create the new one and apply to the Groups.

4. Now, we will add 'Administrator Access' Permissions to the user(Iamuser1) we create.

Groups -Admins-tab<permissions> ---<AttachPolicy>---'select' Administrator Access----<AttachPolicy>

-Dashboard -Customize the IAM link replacing the ID with any name. To Hide the ID need to customize.

IAM user sigin in

https://4234324234.signin.aws.amazon.com/console

After Customize

https://classroomuser.signin.aws.amazon.com/console

- Open the new tab in the browser

https://classroomuser.signin.aws.amazon.com/console

IAMuser =Iamuser1

password=test1234

5. Now need to login using the IAM user, which we created.

Once login , we can launch an EC2 instance.As this user(Iamuser1) is provided with Admin access.

==========

Requirement:

==========

I got an requirement to create a new user and he should be able to do only 'stop' and 'start' , 'reboot' select instances only.

He should not have the permissions to terminate the EC2 Instances.

He should not have the permissions to create the new EC2 Instance.

1. Login to your AWS Console with your root login.

2. IAM -Create another user

    User name* = Iamuser2

    Access Type ='select' "AWS Management Console access"

    'select' CustomPassword ="<somepassword>"

    <NextPermissions>

    Not selecting any group here

    <createuser>

3. Signout and Login using the 'Iamuser2' and its credentials

    Open browser

        https://classroomuser.signin.aws.amazon.com/console

    login with Iamuser2 credentials

    Services ---EC2

    you will get an 'Authorization Error'

4. To view EC2 instances need to provide read permission to the user 'Iamuser2'.

- using Tags, we can provide permissions to this user.

Login using the Root user

EC2 Instances

Select the Running Instance

click on tab <Tags>

add new tag

Key =user

Value=Iamuser2

<save>

5. Using this we can restrict the user to create EC2 instances. We can allow him to do only 'stop' and 'start' Instances.

For this, need to write the custom scripts.

Open the browser search for ='restrict aws user ec2 instance'

https://aws.amazon.com/premiumsupport/knowledge-center/restrict-ec2-iam/

copy the script and open in any editor and customize it.

arn:aws:ec2:us-east-1:111122223333:instance/*"

(Note: For every service we have arn (amazon resource name), but for EC2 there is no arn naming)

InterviewQuestion:If anyone ask you , arn is not displaying for the EC2 instances?

Ans:Simply say that, ARN is not visible for the EC2 instances, but for the other services like S3, we have ARN url.

copy the script

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "ec2:StartInstances",
            "ec2:StopInstances",
            "ec2:RebootInstances"
         ],
         "Condition": {
            "StringEquals": {
               "ec2:ResourceTag/Owner": "Bob"
            }
         },
         "Resource": [
            "arn:aws:ec2:us-east-1:111122223333:instance/*"
         ],
         "Effect": "Allow"
      },
      {
         "Effect": "Allow",
         "Action": "ec2:Describe*",
         "Resource": "*"
      }
   ]
}
```

After Customization

==========


```
          {
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "ec2:StartInstances",
            "ec2:StopInstances",
            "ec2:RebootInstances"
         ],
         "Condition": {
            "StringEquals": {
               "ec2:ResourceTag/user": "Iamuser2"
            }
         },
         "Resource": [
            "arn:aws:ec2:us-east-1:449938344550:instance/*"
         ],
         "Effect": "Allow"
      },
      {
         "Effect": "Allow",
         "Action": "ec2:Describe*",
         "Resource": "*"
      }
```

]

}

==========

Note:

449938344550 = Root AccountID

6. copy the script after customization

IAMUser

Policies ---<createPolicy>---'select' JSON tab

paste the customized script.

<ReviewPolicy>

7. Review Policy

Name ='UserRestrictEC2Instance'

<createpolicy>

8. Now, need to add this policy to the user or groups.

select Users

'Iamuser2' ---Permissions(tab)-- Add Permissions ---AttachExisting Policies directly

Filter policies ='UserRestrictEC2Instance'

Select the policy(UserRestrictEC2Instance') ---<Review>--<AddPermissions>

9. Login to IAM user console

Iamuser2/password

- Now Try to Terminate the EC2 Instance. It throws an error

- Try to Launch an EC2 instance , it throws an error.

  Like this we can restrict the user by creating some policies and apply to it.

  AWS provides the readymade(default) policies we need to customize as per our requirement.

What is IAM ?

What is Root Account ?

How to enable MFA for root account

What is IAM account

How to create IAM account

Programmatic Access Vs Console Access

Attaching Policies to User

Creating Custom Policy

Creating User Group

Adding Users to Group

Adding Policies to User Group

What is IAM Role
