Google Authenticator → used to create a virtual MFA.

→ What is the specific Reason to create a bucket in same region?

A: To reduce latency so that it takes less time to access data. It is also known as storing it in edge locat^{n}

→ Can we create a bucket starting with no.?

A: Yes, we can, 3-63 characters & can start with lower case letter & no. It should not start (or) have any spcl characters (or) uppercase letters.

→ YAML is the other format used in CloudFormat^{n} other than JSON.

→ If we want to update stack then what should be done?

A: By using AWS console / manually by using JSON editor

→ No. of resources in template → 200.

→ It allows customers to manage users & permissions in AWS.

→ Every user we create in the IAM system starts with no permissions

→ When accessing an organisat^{n}, AWS use of API access credentials. Console passwords, Access keys & Signing certificates must be evaluated

→ IAM limit → 2 to 1,224 characters. Alphanumeric without spaces. Can include symbols

→ Path names must begin & end with "/". Forward

Features of IAM:

1. Shared access to your AWS account.
2. Granular permissions
3. Secured access to AWS resources for applicat^n that run on EC2
4. MFA 5. Identity Federat^n 6. Free to use
7. Eventually consistent

* Over 500 resources types are supported by CloudFormat^n covering over AWS services.
* Each cloud Format^n account is limited to a max. 200 stacks.
* limit can be removed by requesting mail.

EC2: Elastic Cloud Compute.

~~AZ: Availability zones:~~

What: This is a webservice from Amazon which provides resizable compute services in the cloud.

Resizable bcz they can easily scale up/down.

Why: Designed to make web-scale cloud computing easier for developers. It's a simple web service interface allows you to obtain & configure capacity with minimal friction.

AMI (Amazon Machine Image): These are templates of OS & they provide the info needed to launch an instance.

Types of EC2 instances: MS

1. General Instances: Balance of performance & cost: eg: email
2. Compute " : lot of processing from CPU. eg: analyzing data from stre
3. Memory " : heavy in nature, require lot of RAM, eg: multitasking of data
4. Storage " : huge in size/have a dataset occupies alot of space
5. GPU " : heavy graphics rendering - eg: 3D modelling.

~~dataset genero~~

Free tier eligible instance is "t2.micro."

t1 & t2 belongs to General instances

From GPU → what is the initials →

IAM, EC2, Networking

AMI: It is an image of a server - including an OS & often additional s/w - which runs on AWS
- Preconfigured templates for creat^n of virtual servers (EC2 instances) in AWS environment.
- users can select an AMI provided by AWS, the user community, (or) through AWS marketplace.

Example for AMI:
If we want to use a hadoop s/w then first we have to select AMI which contain that on it.
Cost of EC2 instance.

What can I do with EC2?
It allows you with complete control of your computing resources.
It reduces time required to obtain & boot new server instances to minutes, scale up/down as your computing requirements change.

Use case for T3 instances (upto 5 Gbps): Microservices, low-latency interactive applicat^{n}s, small & medium dbs, virtual desktops, development environments, code repositories, business-critical applicat^{n}s.
It has burstable CPU.

a) General → A1, T3, T3a, T2, M6g, M5, M5a, M5n, M4.
b) Compute → C5, C5n, C4
c) Memory → R5, R5a, R5n, R54, X1e, X1, High memory, z1d
d) Storage → I3, I3en, D2, H1.
e) GPU → Accelerated Computing → P3, P2, Inf1, G4, G3, F1

1. Identity Federat^{n}: It is a method that uses an existing authentication solut^{n} to grant access & authorizat^{n} to another solut^{n} without recreating user IDs and having multiple passwords to remember

There are several types of authenticat^{n} & several technologies that can be used to authenticate btwn independent solutions with a common & centralized authority source

It comes in many forms but the objective & funct^{n}'s is pretty much the same. Use a central directory to maintain user IDs & passwords that can grant access to an independent solution without having to have multiple user stores.

Common purpose of it is to grant or revoke user access from a single locat^{n} to multiple services. This way, a user can come & go from the organizat^{n}'s You don't have to administer the user accounts & authorizat^{n} in every single system that u have in your enterprise. AWS is simply an extension of your enterprise it should be treated as such so that u can control access to our new infrastructure services at AWS from a single federated source

The concept of this is common & there are several underlying technologies that can be used to achieve this authorizat^{n} from a single central locat^{n}

Jayendra Patil's Blog

* AWS Identity Federation is the concept of using external authorization sources to permit access to AWS console & AWS resources.

* Placement Groups: It determines how instances are placed on underlying hardware. These are logical groupings/clusters of instances in a particular AWS region. Mainly used for launching cluster compute instance types.

3 strategies: Cluster, Partition & Spread.

1. Cluster: Clusters instances into a low-latency group in a single AZ. It can span peered VPCs in same Region. (high bandwidth)
2. Partition: Spreads instances across logical partitions, ensuring that instances in one partition do not share underlying h/w with instances in other partition. (There are 3 partitions).
3. Spread: Spreads instances across underlying h/w. (7 instance in same AZ)

→ A partition placement group can have partitions in multiple AZs in same Region.

Rules: 1. Name should be unique for the region.
2. U can't merge pgs.
3. Only one instance can be launched at a time.

* We can add 50 tags per pg.
* Max 8 clusters per AWS account in cluster(s)

What: Feature that enables EC2 instances to interact with eachother via high bandwidth, low latency connection.

* PGs is only available within a single AZ & across peered VPCs.

* EBS optimizat"s:

EBS supports single instance environments, with a single instance & autoscaling to maintain max/min instance.

⇒ EBS-optimized instances deliver dedicated bandwidth to EBS. These uses an optimized configurat" stack and provides additional, dedicated capacity for Amazon EBS I/O.

a1.medium *	a1.4xlarge
a1.large *	a1.metal
a1.xlarge *	c4.large
a1.2xlarge *	c4.xlarge

max a1 ⇒ 3,500 max
c4 ⇒ 500, 750
min: 437.5 → a1
62.5, 93.75 } c4.

* EBS snapshots:

Backup data on EBS volumes to S3 by snapshots.

⇒ These are incremental backups, which means only blocks on device that have changed after recent snapshot are saved.

⇒ Tracking the status of EBS snapshots can be done through CloudWatch events.

* Pricing: Charges for snapshots are based on amount of data stored.

⇒ It supports EBS encrypt"n

→ If you copy & encrypt it to a new CMK, a completed copy is already created, resulting in additional delay & storage costs

→ A snapshot can is constrained to region where it was created.

→ Can copy snapshots across Regions.

* Security Groups Vs (NACLS) (Network Access Control lists:

→ SG: Acts as firewall for associated Amazon instances (1st layer of defense) controlling both inbound & outbound traffic at instance level

→ NACLS: Acts as firewall for associated subnets, controlling both inbound & outbound traffic at subnet level (2nd layer of defense)

→ SG: Each instance within a subnet can be assigned a diff. set of SG.

* Instance can be assigned 5 SGs with each SG having 50 rules
* Default SG allows no external inbound traffic but allows inbound traffic from instances with same SG. & allows all outbound traffic.
* These can "specify only allow rules, but not deny rules"
* SG are stateful. & associated with ENI (n/w interfaces)

→ NACLS: It is an optional layer of security for the VPC

* These are not for granular control
* Have seperate inbound & outbound rules, each rule can either allow (or) deny traffic.

→ Default ACL Allows all inbound & outbound traffic

→ Newly created ACL denies " " " " "

* NACL is a numbered list of rules that are evaluated in order to start with lowest numbered rule.
* NACLs are stateless.

* IAM users: It is an entity that we create in AWS. Primary use of it is to give people the ability to sign in to Console for interactive tasks & make programmatic request to AWS services using CLI/API.
Default - Deny.
2 ways user authenticates against AWS:
a) API Access key/secret: Machine usage
b) Login/Password - console usage
* In-line policy: A policy that lives in user & can only be used by user i.e., directly attached to the user & is permitted to that exact user.
This cannot be shared with other users, groups/role
* Cannot have more than 2 API key/secret pairs assigned to user.
* User can belong to not more than 10 groups at same time
* Max 5000 users can be created in an account

* IAM Groups: (Collection of users).

Instead of attaching policies to each user directly groups will be formed & centralized authorization matrix for multiple IAM users.

* Policies attached to groud, donot limit but extend policies attached to User.

* If User has full access to S3 & Group has full access to EC2 then that user will have full Access to both EC2 & S3.

* IAM roles: Similar to user. It determines what identity can & cannot do in Aws. Roles donot have any credentials.

* It can be created with some specific permissions.

* Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

* When u assume a role, it provides temporary Security credentials for your role session.

STS: Security Token Service.

It provides short-term credentials.

*IAM Policies : Policy is a set of permissions which apply to a user, group / role.

It contains statements - Each statement contains:

a) Effect - Allow / deny
b) Actn - API calls.
c) Resource - on which resource this statement has an effect
d) Condition - in which circumstances this statement has applied.

6 types of policies

1. Identity based policies < Managed < AWS managed / Customer managed
                            < Inline
2. Resource based policies.
3. Permissions boundaries
4. Organizations SCPs (Service Control Policy)
5. Access Control Lists (ACLs)
6. Session Policies.

JSON Policy Structure: Series of elements.

Version
Statement
Sid (optional)
Effect
Principal
Actn
Resource
Condition (optional)

→ Processes in milliseconds.

AWS Lambda (Serverless Computing).

- With serverless computing, applicat° runs on servers, but all server management is done by AWS.
- Pay per use basis.
- It provides easy scaling & high availability to the code without additional effort on your part.
- It runs many instances in parallel.
- lambda stores code in S3 & encrypts it at rest & performs additional checks while the code is in use.
- Time out is between 1 & 900 sec. Default is 3 sec.
- Event source publishes events & lambda funct° is a custom code that processes events.
- Restrict°s:
  a) Inbound n/w connect°s are blocked by this.
  b) Outbound n/ws only TCP/IP sockets are supported.
- Lambda automatically monitors funct°s, reporting real-time metrics through cloudWatch.
- Lambda automatically integrates with cloudWatch logs.
- Lambda funct° supports code written in:
  - Nodejs
  - Python
  - Java
  - C#
  - Go.

↳ for sensitive info., AWS recommends using client-side encrypt using AWS key managemt service & store resulting values as cipher text in environment variable.

↳ DO versioning of same funct.

↳ Versioning can be completed using Aliases.

• Aliases is a pointer to a specific funct version with unique ARN.

• Aliases helps in rolling out new changes / rolling back to old versions

↳ It can be configured to connect to private subnets in VPC in aws account.

*Def: AWS lambda is a responsive cloud service that inspects actions within applicat & responds by deploying the user-defined codes known as functions

↳ Never declare any funct variable outside the scope of the handler.

Benifits:

a) lambda tasks need not to be registered like amazon SWF activity types.

b) We can use existing lambda functions that have already defined in workflows.

c) Lambda provides us the metrics & logs for tracking function executions.

Limits

* 3 types

a) Throttle limit: 100 concurrent lambda functions per account (can be increased by contacting support).

no. of concurrent executions for a funct^n = (avg. durat^n of funct^n execut^n) x (no. of request or events processed by AWS lambda)

b) Resources limit:

Resource	Default limit
• Ephemeral disk Capacity ("/tmp" space)	512MB
• No. of file descriptors	1024
• No. of processes & threads (combined total)	1024
• Max. execut^n durat^n per request	300 sec.
• Invoke request body payload size	6MB
• Invoke response body payload size	6MB

c) Service limit.

• Lambda funct^n deployment package size (.zip/.jar file)	50MB
• Size of code/dependencies that u zip into deployment package (uncompressed zip/jar size)	250MB
• Tot. size of all deployment packages that can be uploaded	1.5GB
• No. of unique event sources of Scheduled Event source type per account (per region)	50
• No. of unique lambda functions u connect to each scheduled event	5

EC2 Architecture:

EC2 allow users to use virtual machines of diff. config. as per their requirement.