**AWS**

**Amazon web services**

**CONTENTS**

|  |  |
| --- | --- |
| **Chapter** | **Topics** |
| Global Infrastructure | A Conceptual Introduction to Amazon Web Services (AWS)  Regions  Availability Zones  End points  Creating an AWS Account Usage Tracking Billing |
| IAM | IAM Essentials  IAM Policies  IAM Users  IAM Groups  Securing your account – Creating an IAM user and setting up the CLI  IAM Roles |
| Cloud Watch | Alarms  Events  Monitoring  Logging |
| AWS Simple Notification Service SNS | Introduction to SNS  Understudying Topics Subscription and Publisher concepts |
| CloudTrail | Understanding the concepts CloudTrail |
| VPC | Basic understanding of VPC  Subnets  Route Table  Internet Gateway  NAT Gateway  Security in VPC: NACL  VPC Peering  VPC Endpoints  Limitation of VPC |
| Server Based Compute (EC2) Fundamentals | EC2 Architecture  Instance, Types and Sizes  EC2 instance purchasing types  Instance Roles  AMI’s  Ec2 Storage Architecture  EBS, Types  EFS  Overview of Different types of storage classes  Snapshots  Life cycle manager  Security Groups  Elastic IPs  Key pairs  Load Balancing  Autoscaling  Route 53 |

|  |  |
| --- | --- |
| **Chapter** | **Topics** |
| S3 | Understanding S3  S3 Naming Convention  Transferring Data to S3  Storage classes in S3  Lifecycle policy in S3  Cost optimization for S3  Versioning in S3  Encryption in S3  Static website and CORS |

**(1)**

**GLOBAL INFRASTRUCTURE**

1. **Aws Free tier account and monitoring**

Billing dashboard

go to preferences and enable the receive free tier usage alert

1. **Global infrastructure**

We can access AWS i) AWS console ii) AWS cli iii) SDKs

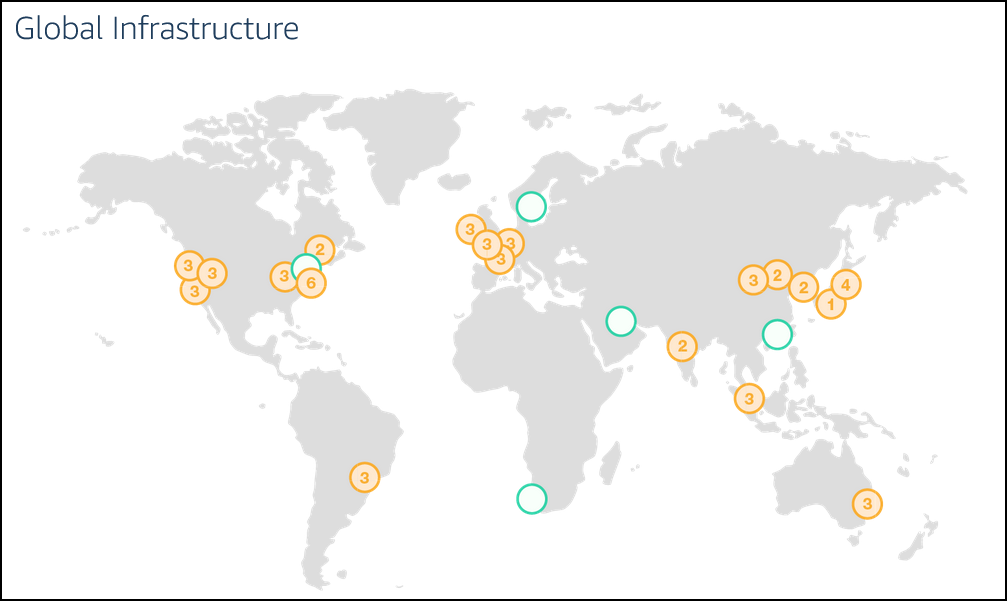
1. **AWS Region:**

* *AWS Region* is a separate geographic area where we cluster data centers.
* Each AWS Region is completely independent.

**Availability zones:** Each **AWS Region** consists of multiple, isolated, and physically separate AZ's within a geographic area.

**Local zone:** A *Local Zone* is an extension of an AWS Region that is geographically close to your users, for low latency communication.

**Edge location:** its data center, where end users access services **located** at AWS to reduce latency



1. **Compute section:**

EC2 Elastic compute cloud: Virtual machine

ECS Elastic Container service: Container as a service.

EBS Elastic beanstalk: PASS solution manages infrastructure for you.

AWS Lambda: Serverless computing platform.

1. **Storage overview:**

**Databases:** SQL Database - RDS, NOSQL Database - Dynamodb, Elastic cache, Redshift.

**Storage:** AWS S3 object storage service file used in application storing file and retrieving files.

**(2)**

**IAM (Identity and Access Management)**

* Provides access to accounts services where we can manage User, Roles, Groups & Policy password policy.
* It applies globally to all AWS regions.

**Users**: we create users and assign necessary permissions to them in the form of policies.

**Groups:** We can create groups for ex. Dev QA etc. and attach policies at the group level.

**Policy:**

A policy is a set of permission

Always explicit deny overrides explicit allow

**{**

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

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Action": \*,**

**"Resource": "\*"**

**},**

**Policy types:**

1. **Identity Based Policy:** Applicable on users, groups of users, and roles

* AWS Managed policy:
* Custom Managed Policy:
* Inline Policy

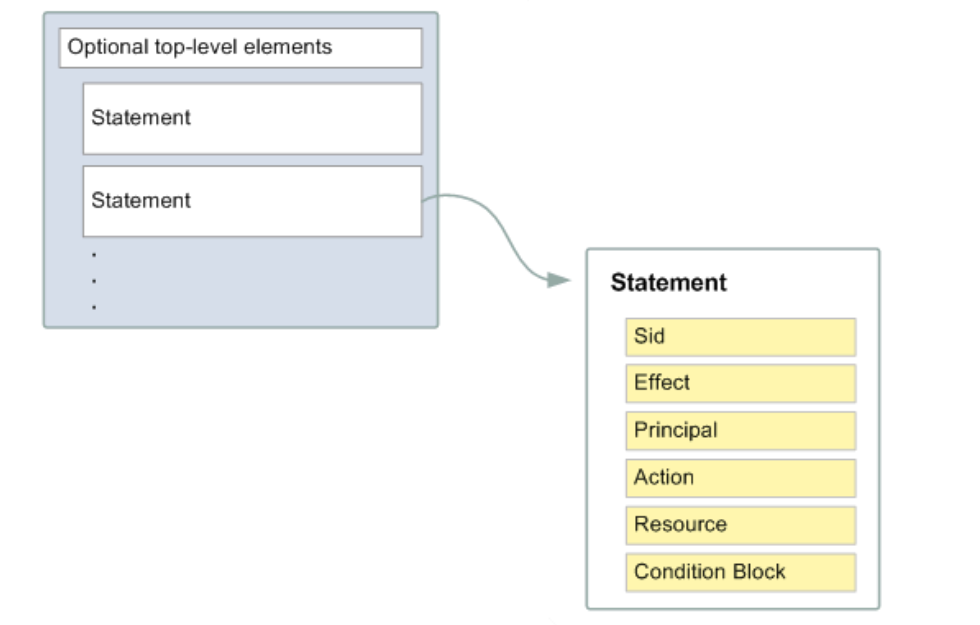
1. **Resource Based policy:** Attach to a resource such as an Amazon S3 bucket
2. **Session based Policy:** create a temporary session for a role or federated user

Imp Notes:

More than one policy can be attached to a user or a group at the same time.

Policies can’t be attached directly to resources like EC2 instance, S3 bucket etc.,

**Basic Policy structure:**



**Effect** : Can take only two value allow or deny

**Principal:** who is assuming the policy

**Resource:** on whom you are assuming the policy

Ex:

{

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

"Statement": [

{

"Sid": "FirstStatement",

"Effect": "Allow",

"Action": ["iam:ChangePassword"],

"Resource": "\*"

},

{

"Sid": "SecondStatement",

"Effect": "Allow",

"Action": "s3:ListAllMyBuckets",

"Resource": "\*"

},

]

}

Q: Can we attach multiple policies to user group or a role.

Yes, we attach.

**Roles:**

* Roles comes between services like ec2 wants to access S3 or non-AWS user (hybrid account) should access AWS Resources.
* Policies can’t be attached to aws resources hence roles come into picture
* EC2 can be attached one role at a time.
* Can we assign multiple roles to a EC2 instance? No, we can’t. we can assign only single role to EC2 instance.

Q: If an ec2 instance is not able to access s3 bucket what could be the reason

A Role needs to be attached with proper policy defined.

**Assume Role:**

Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token.

Example:

1. aws s3api list-buckets

2. Remove s3 access policy from the user

3. Try to list the bucket - you can't list the bucket because you the user is not having the

permission to list bucket (access denied)

4. Grant the user to assume a role

Create a role

Attach a policy s3 full access policy

get inside the role ---- trust Relationship change it to user instead of ec2

AWS: "ARN OF IAM USER"

5. aws sts assume-role --role-arn <enter the role arn> --role-session-name s3-access-example

6. copy the accesskey secretkey and seeison token

7 set AWS\_ACCESS\_KEY\_ID=<enter the copied access key>

set AWS\_SECRET\_ACCESS\_KEY=<enter the copied session key>

set AWS\_SESSION\_TOKEN=<sessiontoken>

8. aws s3api list-buckets

9. Remove the env variable

set AWS\_ACCESS\_KEY\_ID=

set AWS\_SECRET\_ACCESS\_KEY=

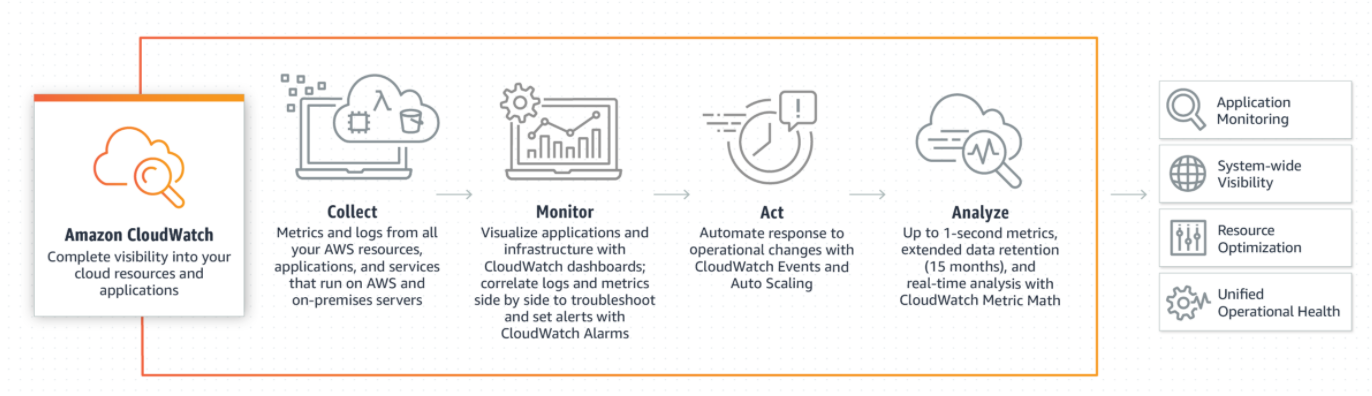
set AWS\_SESSION\_TOKEN=

The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

**(3)**

**CloudWatch**

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time



Default metrics of EC2 instance: Network usage CPU Usage

**Metrics:**

Metrics are data about the performance of your systems

Basic monitoring: which polls for every 5 minutes

Detailed monitoring: which polls for every 1 minute.

**Alarm:**

If CPU utilization goes beyond the static threshold alarm goes to alarm state

Alarm state

Insufficient

OK state

Assignment: set a billing alarm to your account

Events: An Event indicates change in AWS environment

Event Resource: Which resource you want to monitor

Event target: to alert the event change through notifications

Logs:

CloudWatch Logs enables you to centralize the logs from all your systems, applications, and AWS services

Assignment follow the link and try to reproduce the same:

Ex: <https://www.youtube.com/watch?v=F4IE69V-iuw>

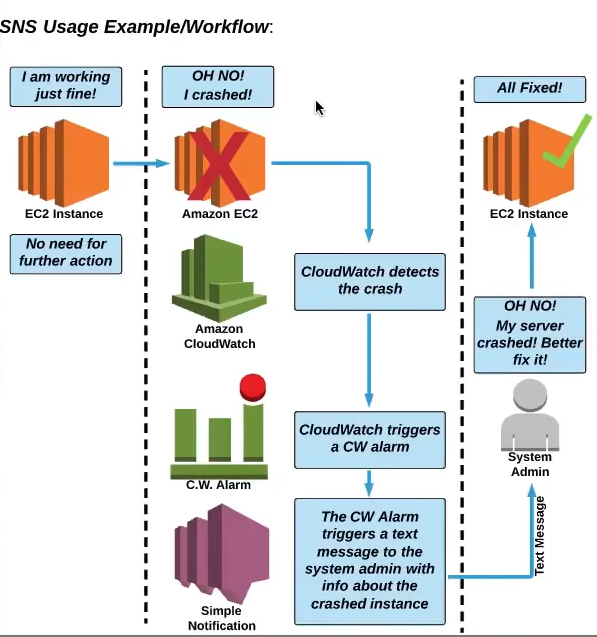
**(4)**

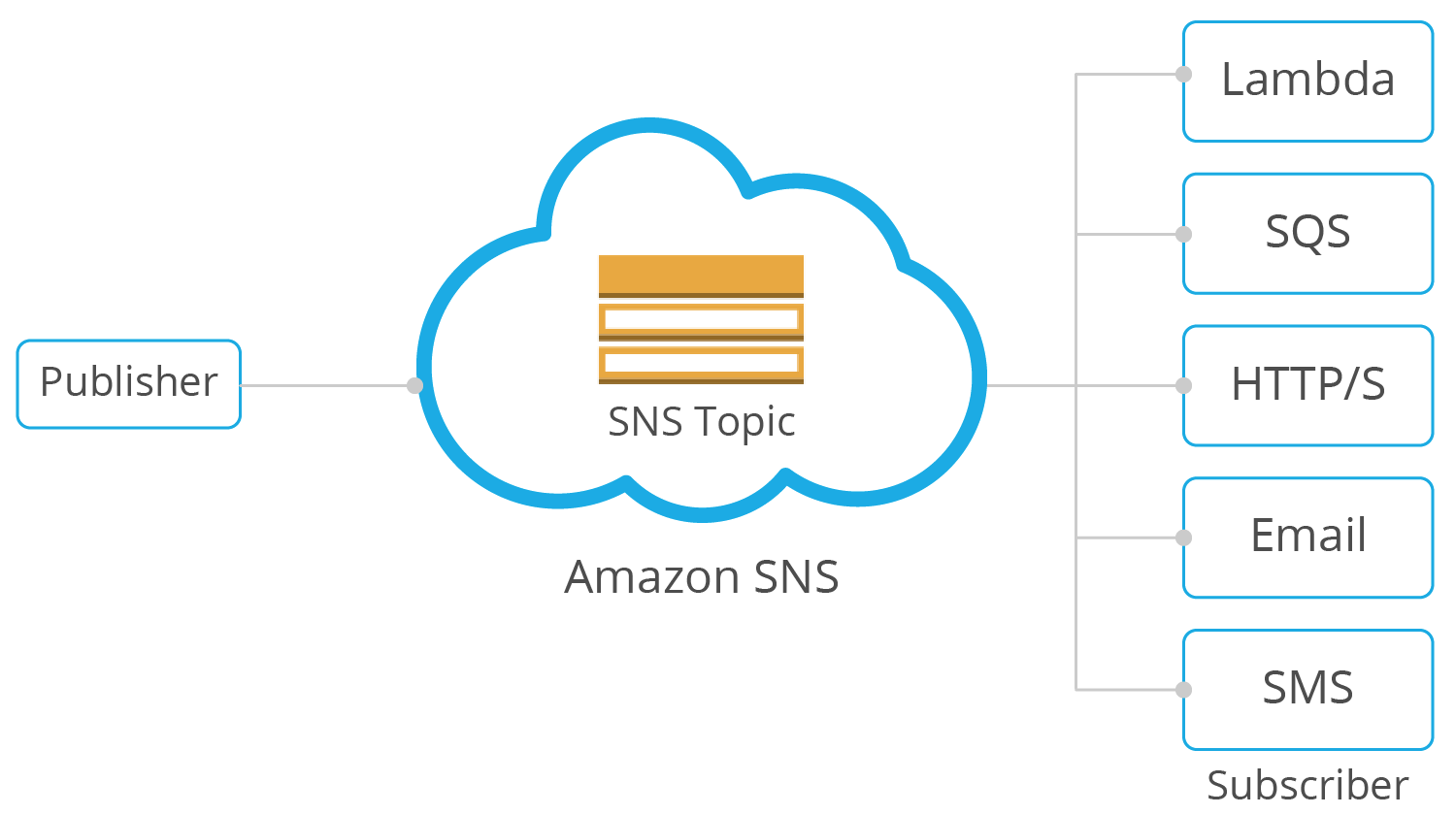
**Simple Notification Service**

* Amazon Simple Notification Service is a notification service provided as part of Amazon Web Service.
* It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users

**Topic:**

An Amazon SNS topic is a logical access point that acts as a communication channel





**(5)**

**CloudTrail**

* Auditing tool records all was account activity.
* Any action taken by users, roles and AWS services are recorded to cloud trial.
* Cloud trial events are kept for 90 days in event history
* You can create a trail of your own store the event history in s3 bucket.
* There are two types of event

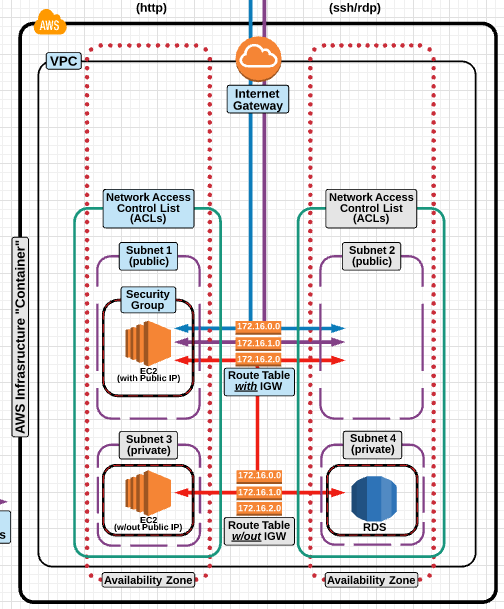
**Management events:** Management operations performed on AWS

**Data events :** currently supported S3 and Lambda: You can now record all API actions on S3 Objects and receive detailed information such as the AWS account of the caller, IAM user role of the caller, time of the API call, IP address of the API, and other details

**(6)**

**Virtual Private Cloud**

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define like EC2 instance Databases.



**CIDR:(Classes interdomain routing)**

Classless Inter-Domain Routing is a method for allocating [IP addresses](https://en.wikipedia.org/wiki/IP_address) and for [IP routing](https://en.wikipedia.org/wiki/IP_routing).

Ex: The IPv4 block 192.168.100.0/22 represents the 1024 IPv4 addresses from 192.168.100.0 to 192.168.103.255.

I.e. 2^ (32-22) = 2^10 = 1024 IPv4 addresses.

**VPC design:**

|  |  |  |  |
| --- | --- | --- | --- |
| VPC CIDR = 10.180.0.0/16 means we have 65536 IPv4 address | | | |
| IPV4 Address range is 10.180.0.0 ---- 10.180.255.255 | | | |
| Public subnet 1 | Public Subnet 2 | Private Subnet 1 | Private Subnet 2 |
| 10.180.0.0/24 | 10.180.1.0/24 | 10.180.2.0/24 | 10.180.3.0/24 |
| 256 IPV4 address | 256 IPV4 address | 256 IPV4 address | 256 IPV4 address |
| 10.180.0.0-10.180.0.255 | 10.180.1.0-10.180.1.255 | 10.180.2.0– 10.180.2.155 | * + 1. – 10.180.3.255 |

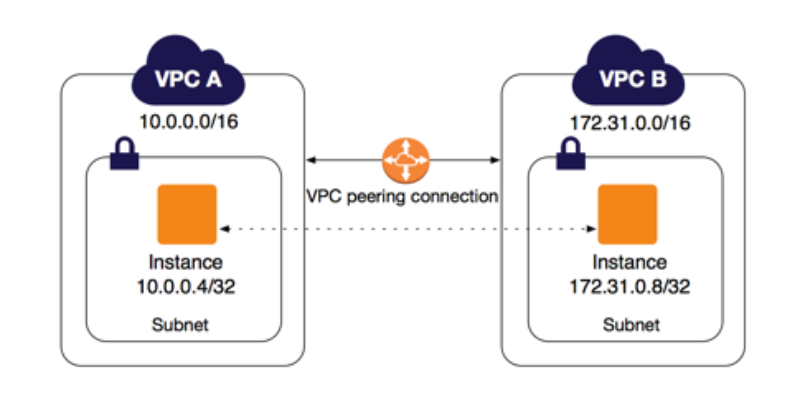
* A **subnet** is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet.
* Some IP addresses are reserved they are
* 10.180.0.0 Network address
* 10.180.0.1 VPC Router
* 10.180.0.2 DNS server (**DNS**. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses)
* 10.180.0.3 Future use
* 10.180.0.4 N/W Broadcast address
* VPC spans multiple Availability zones.
* Subnets must be associated with route table
* A public subnet has a route to internet
* A private subnet doesn’t have route to internet. It creates higher level of security.
* You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances.

**VPC Quota or VPC limitations**

* 5 VPC per region
* 5 IGW per region
* Subnet per VPC 200
* IPv4 CIDR blocks per VPC 4
* Elastic IP addresses per Region 5
* Internet gateways per Region 5
* NAT gateways per Availability Zone 5
* Network ACLs per VPC 200
* Rules per network ACL 200

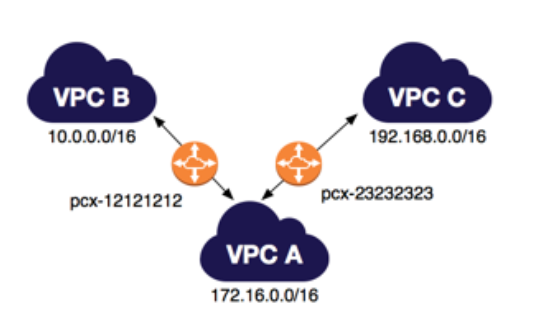
**VPC Peering**:

* A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.
* Instances in either VPC can communicate with each other as if they are within the same network.
* You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.
* The VPCs can be in different regions (also known as an inter-region VPC peering connection).

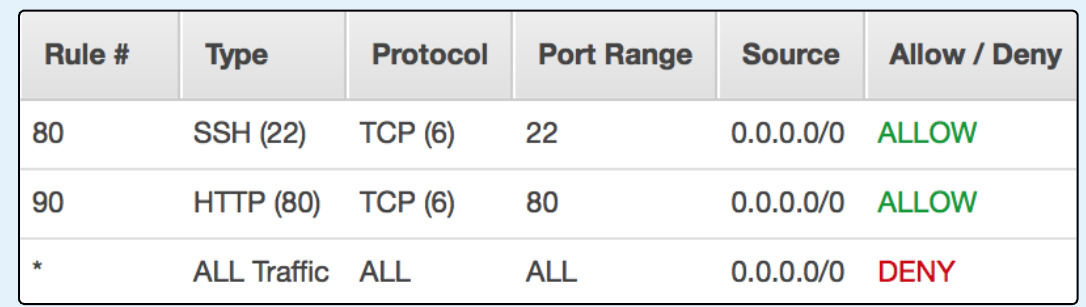


**Conditions:**

* CIDR block shouldn’t overlap
* Transitive peering relationships are not supported. i.e here VPC B cannot connect with VPC C.
* If the VPCs are in different regions, inter-region data transfer costs apply.
* You cannot have more than one VPC peering connection between the same two VPCs at the same time.

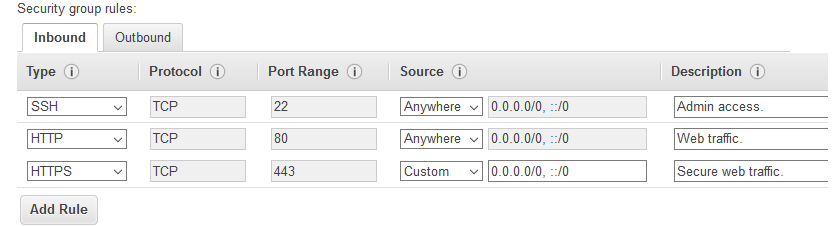


**NACL:**

1. A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. (Firewall at subnet level)
2. Inbound means – incoming (Ingress)
3. Outbound means – outgoing (egress)
4. Always explicit deny take precedence over allow

**Security Group:**

A **security group** acts as a virtual firewall for your instance to control inbound and outbound traffic.

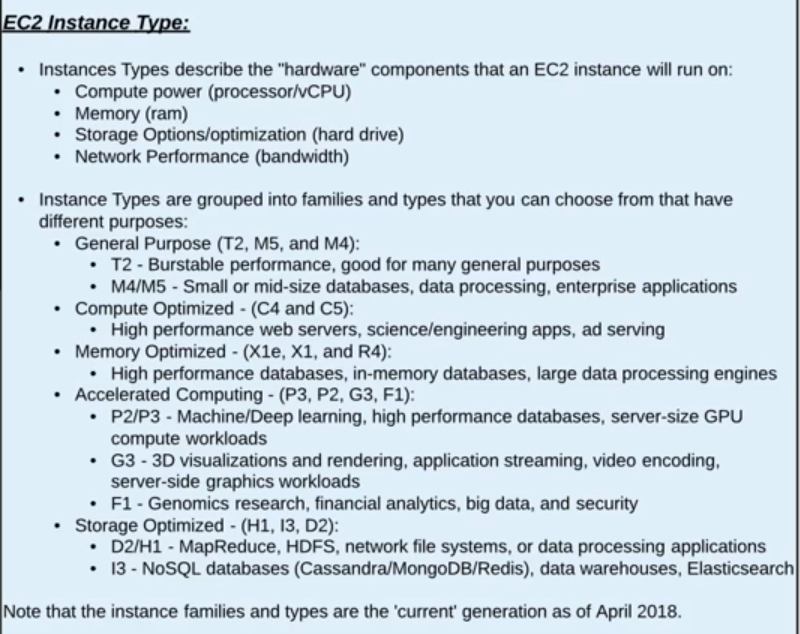


**(6)**

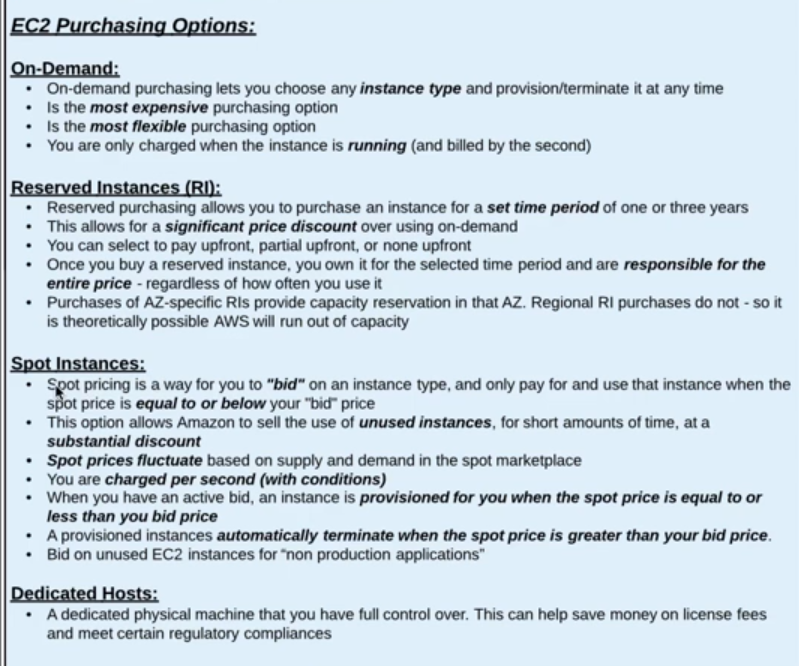
**Elastic Cloud Compute**

An **EC2 instance** is a virtual server in Amazon's Elastic Compute Cloud (**EC2**)

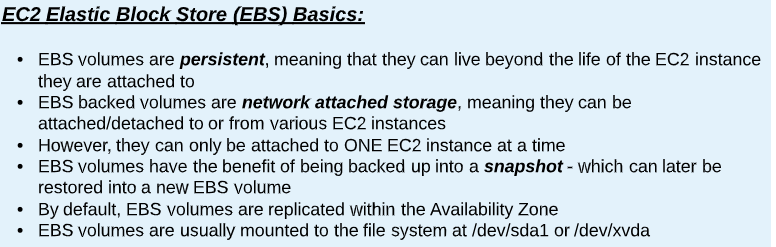
**EC2 instance types:**

****

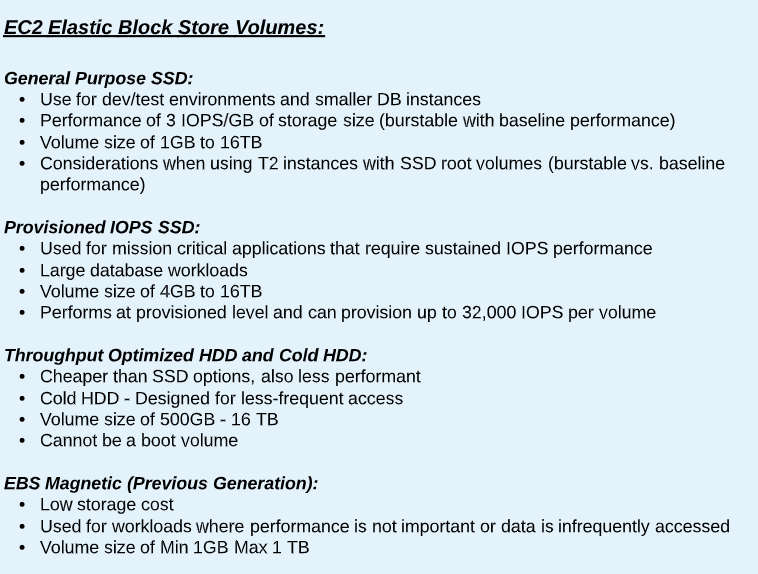
**EC2 purchasing options:**

****

**EBS (Elastic block storage)**

****

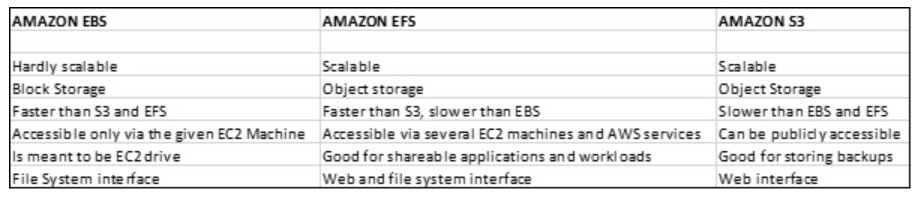
**EBS Types:**

****

**EFS:**

Amazon Elastic file systemis a regional service storing data within and across multiple Availability Zones (AZs) for high availability and durability

**Difference between EBS v/s EFS v/s S3**



**Snapshot EBS**

* You can back up the data on your Amazon EBS volumes to Amazon S3 by taking point-in-time snapshots. Snapshots are incremental backups

**Snapshots are stored in S3**

* Launch two ec2 instance in different az’s(instance1 & instance2)
* Create EBS volume and attach it to instance1
* The volumes are attached to instance1 you can verify it by logging into instance1 and executing “lsblk” command, but it’s not mounted you can verify it through by running command “df -TH”
* Mount the volume to instance1
* Format the disk with ext4: “mkfs -t ext4 /dev/xvdf”
* Create a directory in root: 1. “cd /” 2. “mkdir /mnt/mydisk”
* Mount the disk: “mount /dev/xvdf /mnt/mydisk”
* you can verify that disk is mounted by running “df -TH” command.
* Create some files
* Take a snapshot
* Unmount the disk
* umount /mnt/mydisk
* Detach the volume from ec2 instance.
* delete the volume
* Create a new volume from snapshot
* Attach the volume to newly created instance2.
* Mount the volume to instance2
* Create a directory in root: 1. cd / 2. mkdir /mnt/mydisk
* mount /dev/xvdf /mnt/mydisk

Assignment: Difference between instance level snapshot and volume snapshot

**Data life cycle Manager:**

You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of snapshots taken to back up your Amazon EBS volumes

**Amazon machine image (AMI):**

* An Amazon Machine Image (AMI) provides the information required to launch an instance.
* You can launch multiple instances from a single AMI when you need multiple instances with the same configuration.

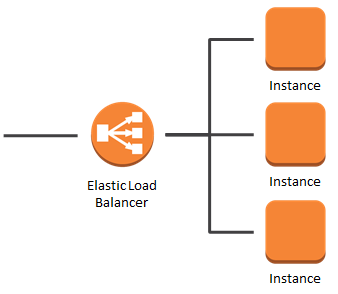
**Difference between Snapshot and AMI**

An EBS snapshot is a backup of a single EBS volume. The EBS snapshot contains all the data stored on the EBS volume at the time the EBS snapshot was created.

An AMI image is a backup of an entire EC2 instance. Associated with an AMI image is EBS snapshots. Those EBS snapshots are the backups of the individual EBS volumes attached to the EC2 instance at the time the AMI image was created.

**Elastic load Balancer (ELB):**

Manage and control the flow of inbound request to group of targets by distributing the requests evenly across the targets. The targets may be EC2 instances lambda or containers.



**Types of Load balancer:**

Application load balancer:

* Used mainly for web application running http and https protocols.
* Operates at request level.

Network Load balancer:

* Ultra-high Performance at very low latency.
* Operates at connection level, routing traffic to targets with in VPC.
* Can handle millions of requests per second.

Classic load Balancer:

* Used for applications that were built in existing EC2 classic env.
* Operates both at connection & request level.

**Example: Classic load balancer**

* Spin up an EC2 instance1 in another availability zone (az1) with http port open in Security group
* Add the below script and launch the instance.

#!/bin/bash

yum update -y

yum install httpd -y

echo '<h1> Resonse from server-1 </h1>' > /var/www/html/index.html

service httpd start

chkconfig httpd on

* Spin up one more EC2 instance1 in another availability zone (az2) with http port open in Security group
* Add the below script and launch the instance.

#!/bin/bash

yum update -y

yum install httpd -y

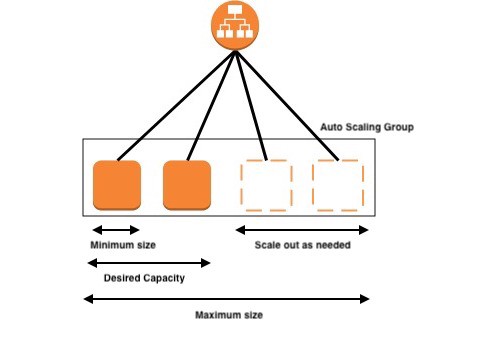
echo '<h1> Resonse from server-2 </h1>' > /var/www/html/index.html

service httpd start

chkconfig httpd on

**Autoscaling**

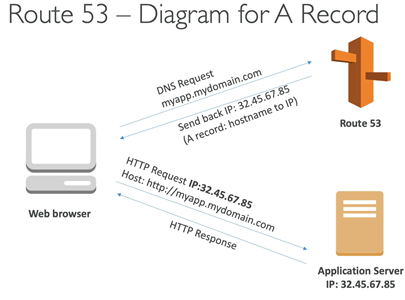
AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.



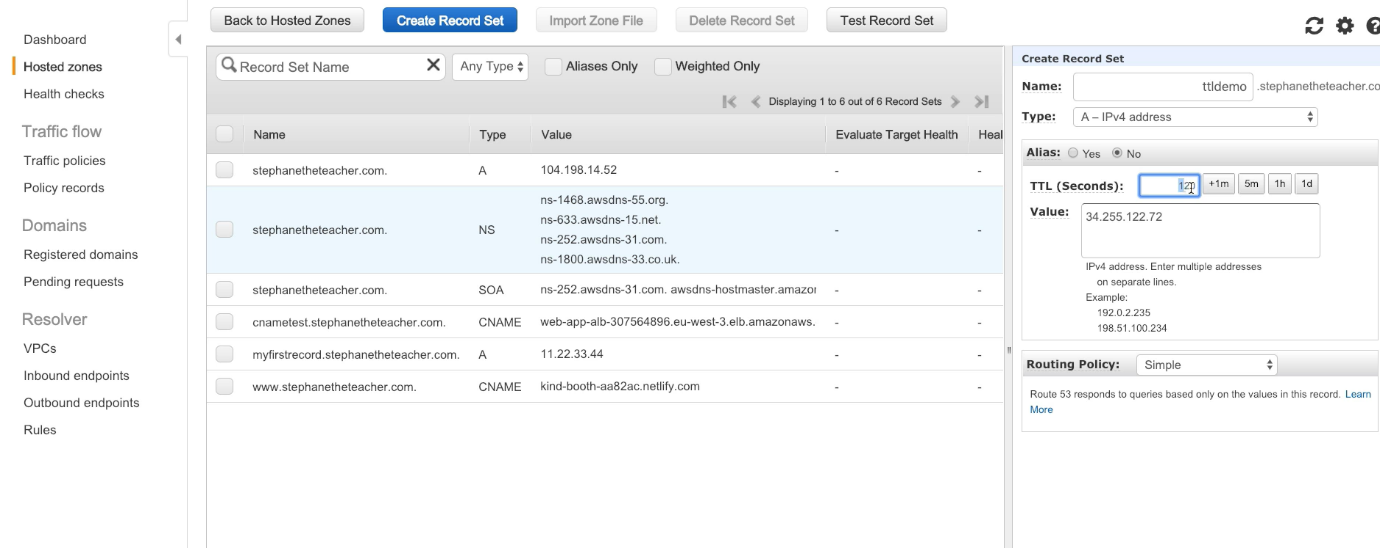
**(7)**

**Route53:**

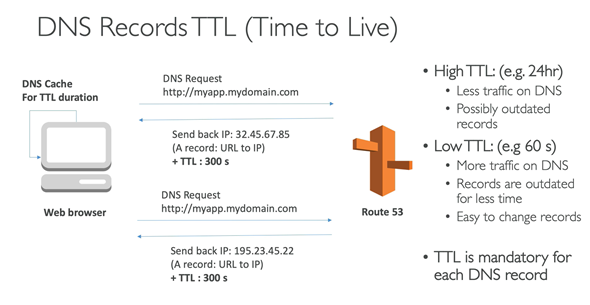
Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service, where we can point IP address to domain name or point host name to another host name.

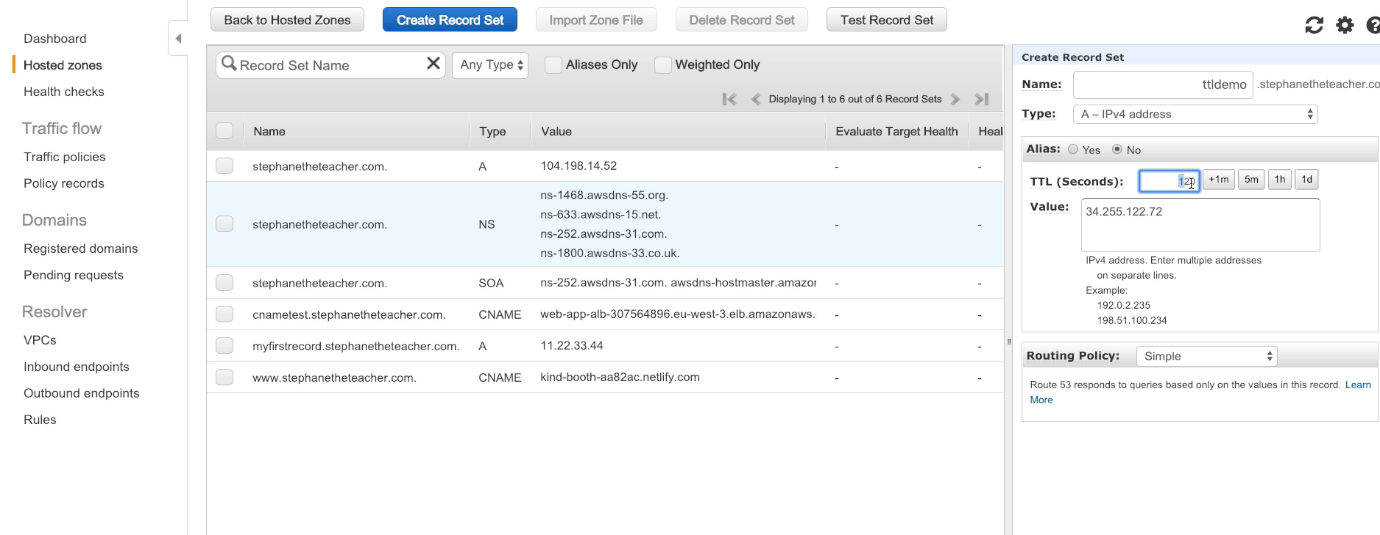
****

**A Record:** Maps IP address to domain name ex: 10.180.0.0 to myapp.mydomain.com



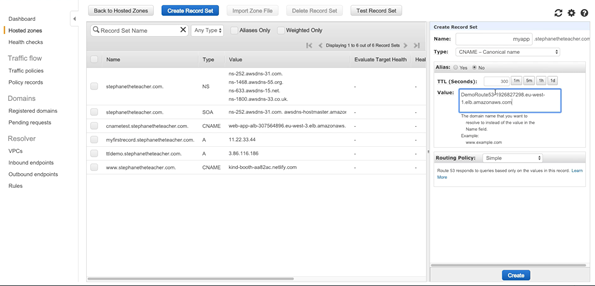
TTL

****

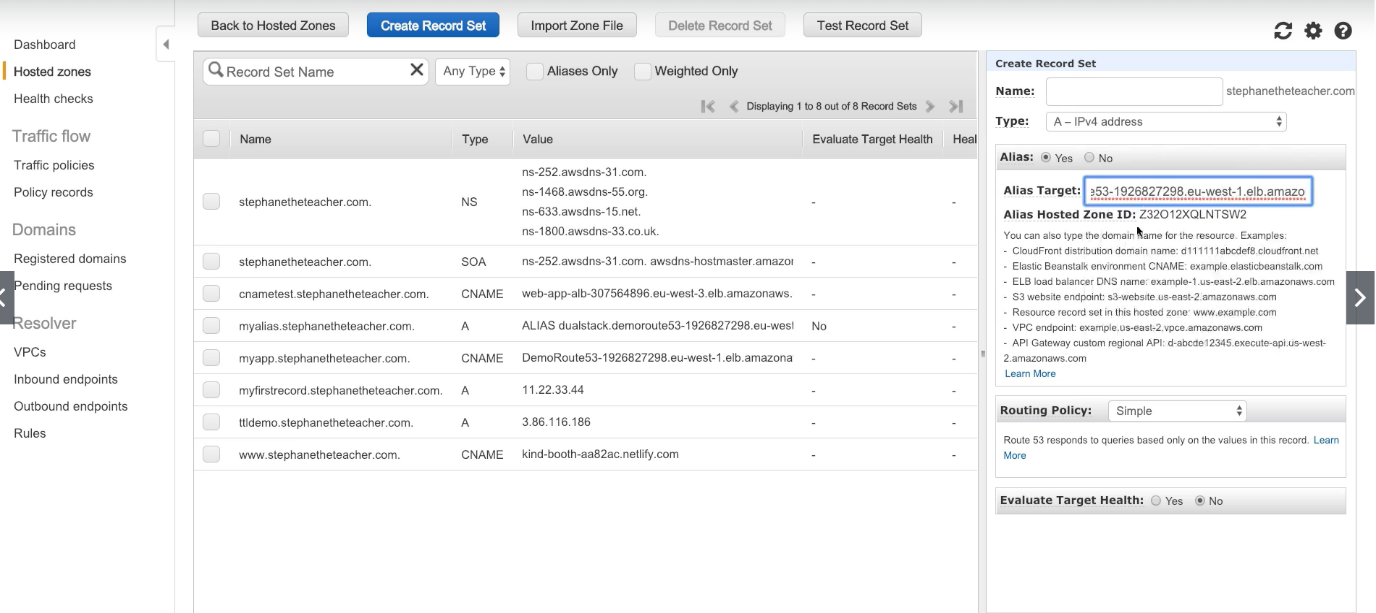


**CNAME Record:**

Maps hostname to another host name: us-east.2.elb.amazonaws.com to myapp.mydomain.com



**Alias Record**: points a host name to AWS Resource ex: myapp.mydomain.com to us-east.2.elb.amazonaws.com



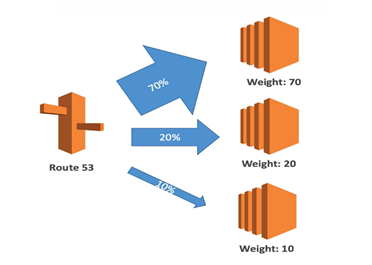
**Latency Routing Policy:**

Use when you have resources in multiple AWS Regions, and you want to route traffic to the region that provides the best latency.



**Weighted Routing Policy:**

Use to route traffic to multiple resources in proportions that you specify.



**(8)**

**Simple storage service (S3)**

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.

**Single operation upload:**

* It’s a traditional upload where you will upload the object in one part
* A single operation upload can upload the file up to 5GB in size.

**Upload object in parts:**

* Using multipart upload, you can upload the large objects up to 5TB.
* You can use multipart upload for the objects from 5MB to 5TB in size.

**Rules for bucket naming:**

* Bucket names must be between 3 and 63 characters long.
* Bucket names can consist only of lowercase letters, numbers, dots . and hyphens -.
* Bucket names must begin and end with a letter or number.
* Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
* Bucket names can't begin with xn-- (for buckets created after February 2020).

**Limitation of S3 bucket:**

* Only 100 buckets can be created per account.
* Can hold unlimited objects

**S3 Storage classes:**

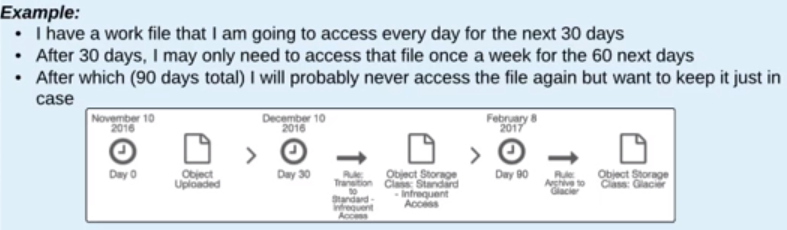
* Standard:
* Designed for general- and all-purpose storage
* Default storage option
* 99.999999999% object durability
* 99.99% object availability
* Most expensive storage class.
* Reduced Redundancy storage
* Designed for non-critical objects
* 99.99% object durability
* 99.99% object availability
* Less expensive than standard
* Infrequent access
* Designed for less frequently accessed objects.
* 99.999999999% object durability
* 99.99% object availability

Less expensive than reduced redundancy storage

* Glacier
* Designed for long term archival storage
* May take several hours to retrieve the objects from this storage
* Cheapest s3 storage class

**S3 Life cycle policy:**

An object lifecycle policy is a set of rules that automate the migration of the object storage class to different storage class

By default, lifecycle policies are disabled for a bucket

**S3 Encryption:**

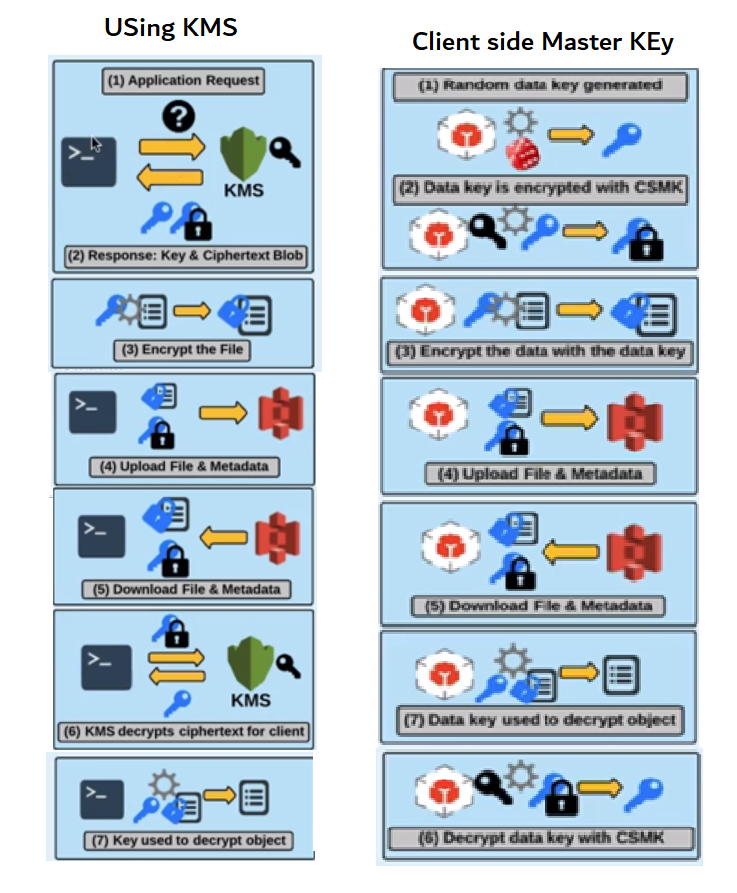
**Two ways of protecting information with S3**

1. **Server side/At rest:**

****

1. **In-transit/Client-side encryption:**

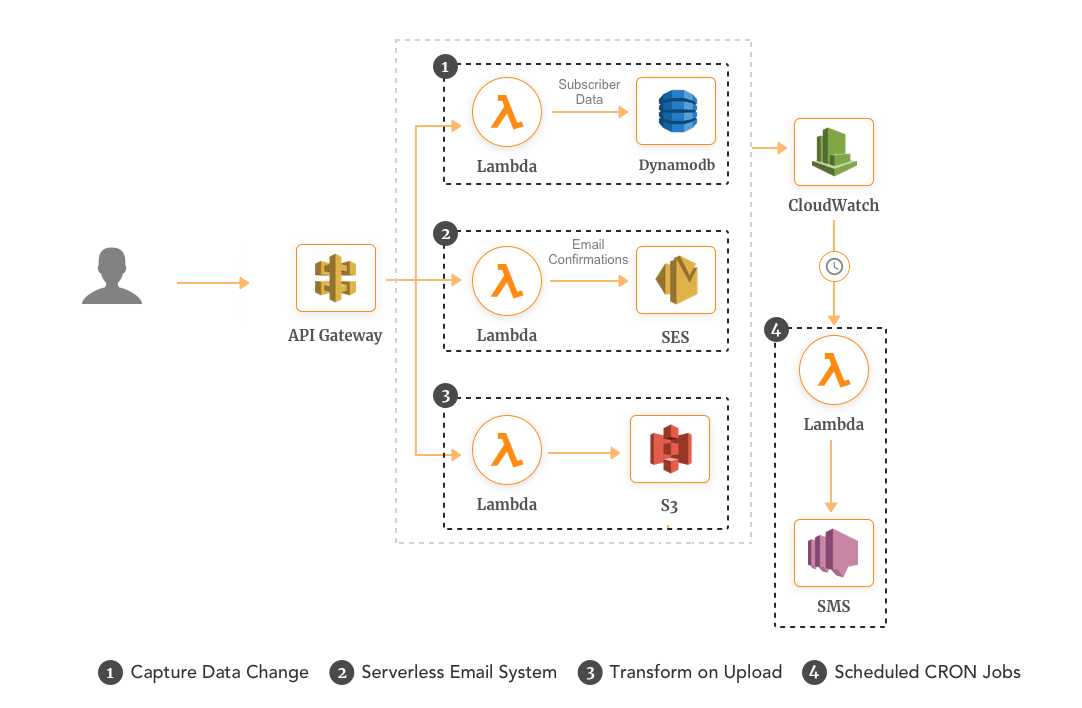
****

****In client side/ in-transit we have two types

**(9)**

**Lambda**

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

 Just upload your code and Lambda takes care of everything required to run and scale your code with high availability

* Manage your virtual functions not really caring about the servers
* Run on demand
* Scaling is automated

**Billing:**

Pay per request first one million requests is free $0.20 per one million request.

compute time 0.00001667 for every GB-seconds used.

**AWS Lambda Languages:**

NodeJS, Python, Python3, Gr00vy, java, csharp, Scala and GO

**AWS Lambda Integration**

Kinesis, API Gateway, DynamoDB, AWS S3, CloudWatch Events, CloudWatch logs, SNS

and Incognito

**(10)**

**Key Management Service (KMS)**

**AWS** Key Management Service (**AWS KMS**) is a managed service that makes it easy for you to create and control customer master keys (CMKs), the encryption keys used to encrypt your data.

**(11)**

**AWS secrets**

AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

**(12)**

**Relational Database Service (RDS)**

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

**Which relational database engines does Amazon RDS support?**

**Amazon RDS database engines:**

* Amazon Aurora
* PostgreSQL
* MySQL
* MariaDB
* Oracle
* Microsoft SQL Server

**Encryption in RDS:**

Encryption at rest is supported for

* Amazon Aurora
* PostgreSQL
* MySQL
* MariaDB
* Oracle
* Microsoft SQL Server

**Q: Can we enable encryption on exciting DB**

Encrypting existing DBs is not supported. To do this, you’ll need to create a new

encrypted instance and migrate data to it. The encryption key can be stored in KMS.

Q: **Which is the non-relational database supported in AWS**

Amazon DynamoDB is the NoSQL database supported by AWS