AWS Solutions Architect Exam notes – 2018

(Associate and Professional levels)

Document Log

Issue	Date	Comment	Author/Partner
0-1	13/07/2018	Almost all the information showed on this document was taken from the following links: http://jayendrapatil.com/ https://aws.amazon.com/documentation/?nc1=h_ls	Jayendra Patil AWS Documentation
1	16/08/2018	Made a summary with all the relevant information taken from: • Jayendra's blog • AWS documentation Also, added some extra needed knowledge for the exam and a key notes paragraph with the main information to know for each relevant service.	Esteban Freire Garcia

Main links used to write this document:

- Almost all the Most the information showed on this document was taken from the following links:
  - http://jayendrapatil.com/
  - https://aws.amazon.com/documentation/?nc1=h_ls

Information to have in account

- AWS Infrastructure and Services features changes every week, so the information of this document probably will need to be updated by the time you read it :)

Table of contents

Refresh some Network knowledge before stating with VPC

How to calculate a Subnet/Network Mask:

Network Classes:

- Class A network (WAN) ***Wide Area Network:**
  - From 1 to 126
  - Network Mask: 255.0.0.0 /8
- Class B network (MAN) ***Metropolitan Area Network**:
  - From 128 to 191
  - Network Mask: 255.255.0.0 /16
- Class C network (LAN) ***Local Area Network (Red Privada):**
  - From 192 to 223
  - Network Mask: 255.255.255.0 /24
- Bloques reservados
  - 10.0.0.0/8 – Private Network
  - 172.16.0.0/12 Private Network-

To calculate the mask, you divide between 2, **where /16 is 65536 (the maximum AWS subnet) and /28 is 16 (the smallest AWS subnet)** . For example /16 is **255.254**, /17 is 256/2=128 and therefore **127.254**, /18 is 128/2 and therefore **63.254** and so on (64/2=32 | 32/2=16 | 16/2=8 | 8/2=4 | 4/2=2 | 2/2=1) until **/24** which is **1.254** (Class C). Remember that you cannot use the first 2 IPs and the last one. 0 is for Network, 1 for router/switch and 255 or the last IP for each subnet, is the broadcast. **For example for 192.168.0.0 (Network):**
/16 = 192.168.0.1 - 192.168.255.254 (65536 IPS)
/17 = 192.168.0.1 - 192.168.127.254
/18 = 192.168.0.1 - 192.168.63.254
/19 = 192.168.0.1 - 192.168.31.254
/20 = 192.168.0.1 - 192.168.15.254
/21 = 192.168.0.1 - 192.168.7.254
/22 = 192.168.0.1 - 192.168.3.254
/23 = 192.168.0.1 - 192.168.1.254
/24 = 192.168.1.1 – 192.168.1.254 (256 IPS)
/25 = 192.168.1.1 - 192.168.1.126 (128 IPs)
/26 = 192.168.1.1 - 192.168.62 (64 IPS)
/27 = 192.168.1.1 - 192.168.1.30 (32 IPs)
/28 = 192.168.1.1 - 192.168.1.14 (16 IPS)

Some AWS tips and tricks:

- In AWS, the **smallest** subnet Classless Inter-Domain Routing (CIDR) block you can have is **/28 netmask=16IPs** and **maximum /16 netmask=65,536 IPs.**

- AWS reserves 5 IPs in each subnet, the first 4 and the last one, for example the 10.0.0.0 is the base network, the 10.0.0.1 the VPC router, the 10.0.0.2 DNs related, 10.0.0.3 reserved for future use and the 10.0.0.255 last IP (broadcast) .**AWS suggests that your account for up to 8 IP addresses in an AZ for scaling up**
  - Example: 10.0.0.0**/26** which extends from 10.0.0.0 through to 10.0.0.63 can be split in two subnets **/27**, 10.0.0.0-31 and then the 10.0.0.0.32 to 63
  - If the VPC is 10.0.0.0**/27 (30IPs)** which extends from 10.0.0.1 to 10.0.0.30 and we have 2 Azs, you must split it in two **/28 subnets**, 10.0.0.1-14
  - Example: In a **/28** subnet in a VPC, you actually have only 11 IPs available for instances and ELB
  - If in the exam it is mentioned a /27 Netmask, it means the subnet only have 30 IPs available and having in account that AWS take 5 Ips (the first 4 and the last one), it could happen that there is not available Ips if there is an ELB configured

Architecture

If you want Multi AZ in your architecture, the minimum is to have your infrastructure deployed over two AWS Availability Zones

- A Three tier architecture is a Web Server (Public Subnet), an APP Server (Public Subnet) and DB Server (Private subnet)
- A Two tier architecture is a Web Server ou AppsServer (Public subnet) and DB server (Private subnet)

About Scaling

- **Vertical Scaling:** Increase individual resources as HDD, Faster Cpu, etc.
- **Horizontal Scaling:** Increase number of resources

(!) DR means Disaster Recovery

Ports and Protocols to have in mind for the exam

Port	Name	Description
7/tcp	echo	Protocolo(Eco)
20/tcp	ftp-data	File Transfer Protocol - data
21/tcp	**ftp**	**File Transfer Protocol - control**
22/tcp	**ssh**	
23/tcp	**telnet**	
25/tcp	smtp	Simple Mail Transfer Protocol
42/tcp	nameserver	

66/tcp and udp	Oracle SQLNet	Remote data access between programs and the Oracle Database.
69/udp	tftp	Trivial File Transfer Protocol
80/tcp	**http**	**HyperText Transfer Protocol**
88/tcp	Kerberos	
115/tcp	sftp	
143/tcp	imap	4 Internet Message Access Protocol ()
389/tcp	ldap	
443/tcp	**https**	
445/tcp	microsoft-ds	Microsoft-DS (Agobot)
465/tcp	smtps	
514/udp		
587/tcp	smtp	
993/tcp	imaps	
1337/tcp		Normally, it is used on compromised/infected servers
1521/tcp		
3128/tcp		
3306/tcp		
3389/tcp		**() Terminal Server**
5432/tcp		
8080/tcp		

VPC

- A virtual private cloud (VPC) is a virtual network dedicated to the AWS account. It is logically isolated from other virtual networks in the AWS cloud.
- VPC allows the user to select IP address range, create subnets, and configure route tables, network gateways, and security settings.
- VPC allows you to set **tenancy** option for the Instances launched in it. By default, the tenancy option is shared. If dedicated option selected, all the instances within it are launched on a dedicated hardware overriding the individual instance tenancy setting
- The public subnet must be in a VPC that has an IGW attached to it, in order for an instance to be access or access the Internet and the subnet's associate route table must have an entry pointing at the IGW for Default Route (0.0.0.0/0)
- You can only configure a IGW per VPC
- A VPC subnet can be private that can access the internet via NAT instance/ gateway (like in a HOME network)
- VPN-only subnet only can be accessed through your Data Center. If a subnet doesn't have a route to the internet gateway, but has its traffic routed to a Virtual Private Gateway (VPG) for a VPN connection, the subnet is known as a VPN-only subnet. (Basically, it connects the Private AWS Subnet from the VPC with your Corporate Network (your on-premise Data Center) using a Virtual Private Gateway (VPG) and through a VPN connection.
- It is not possible to modify the VPC CIDR block after the VPC has been created, and if you need to increase the size you need to create a new VPC. Architects needs to design the CIDR block's sizes carefully and take into account scalability and growth needs and the same for the subnet with the difference you can delete the subnet and create a new one. From August 2017, Amazon Virtual Private Cloud (VPC) now allows customers to expand their VPCs **by adding secondary IPv4 address ranges** (CIDRs) to their VPCs. Customers can add the secondary CIDR blocks to the VPC directly from the console or by using the CLI after they have created the VPC with the primary CIDR block. Similar to the primary CIDR block, secondary CIDR blocks are also supported by all the AWS services including Elastic Load Balancing and NAT Gateway.
- Each VPC is separate from any other VPC created with the same CIDR block even if it resides within the same AWS account
- VPC allows VPC Peering connections with other VPC within the same or different AWS accounts
- Connection between your VPC and corporate or home network can be established, however the CIDR blocks should be not be overlapping.
- Deletion of the VPC is possible only after terminating all instances within the VPC, and deleting all the components with the VPC for e.g. subnets, security groups, network ACLs, route tables, Internet gateways, VPC peering connections, and DHCP options

- The subnet route table by default has a route of VPC CIDR as a destination (TARGET) as Local, so all subnets within a VPC can communicate with one another
- You can see a route table as a virtual router to connect subnets between AZs and to indicate if the output goes through an IGW or through a Virtual Private Gateway (to connect a VPC with office's enterprise)
- The Route table entry to allow instances in VPC to communicate with one another looks like this:
    - **Destination: 10.0.0.0/16 and Target: Local**

Default VPC Components

When AWS creates a default VPC, it does the following to set it up for you:

- Create a VPC with a size /16 IPv4 CIDR block (172.31.0.0/16). This provides up to 65,536 private IPv4 addresses.
- Create a size /20 default subnet in each Availability Zone. This provides up to 4,096 addresses per subnet, a few of which are reserved for our use.
- Create an and connect it to your default VPC.
- Create a main route table for your default VPC with a rule that sends all IPv4 traffic destined for the internet to the internet gateway.
- Create a default security group and associate it with your default VPC.
- Create a default network access control list (ACL) and associate it with your default VPC.
- Associate the default DHCP options set for your AWS account with your default VPC
- DNS resolution

VPC Wizard (From the the console and VPC Dashboard tab)

- When creating a VPC with the Wizard configurations and selecting **“With a Single Public Subnet”,** two route tables are created by AWS for you. The main VPC route table, which gets assigned to the Private IPv4 subnet and a Custom route table, which gets assigned to the public IPv4 subnet. The VPC will have an Internet Gateway (IGW) created and attached
- When creating a VPC with the Wizard configuration and selecting **“With Public and Private subnets”**, the Private subnet, associated with the main route table, will have a NAT gateway (you can change it during the wizard configuration to a NAT instance, if you want a NAT instance, you have to select it by hand during the wizard configuration) to access the internet. In addition to containing a public subnet, this configuration adds a private subnet whose instances are not addressable from the Internet. Instances in the private subnet can establish outbound connections to the Internet via the public subnet using Network Address Translation (NAT). The VPC will have an Internet Gateway (IGW) created and attached
- The NAT instance type by default is **m1 small** (you can choose otherwise but not less than m1 small)
- When creating a VPC with the Wizard configuration and selecting **“With Public and Private subnets and hardware VPN access”**, the configuration adds an IPsec Virtual Private Network (VPN) connection

between your Amazon VPC and your data center - effectively extending your data center to the cloud while also providing direct access to the Internet for public subnet instances in your Amazon VPC.

- When creating a VPC with the Wizard configuration and selecting **“With Private subnet only, and hardware VPN access”,**your instances run in a private, isolated section of the AWS cloud with a private subnet whose instances are not addressable from the Internet. You can connect this private subnet to your corporate data center via an IPsec Virtual Private Network (VPN) tunnel.
  - Configuring a VPC using the VPC wizard that has a VPN-Only and VPN access implies:
    - No public subnet
    - No NAT instance/gateway
    - It will create a virtual private gateway (VGW) but without an Elastic IP
    - It will create a VPN connection
- VGW
  - VPN connection is defined and created
  - You can delete this VPC, everything will be deleted except VGW and VPN connection
  - But the VGW will be detached from the VPC when you delete the VPC
  - VPN connection will still be using the VGW
  - To delete the VGW you have to delete the VPN connection first

VPC Custom (From the console and“Your VPCS” tab)

A custom VPC (created by hand for you) it has:

- 1 route table with **local target** to connect local VMS in the same VPC
- 1 default ACL created
- 1 default Security Group created with source the same group to connect local VMS in the same VPC
- No subnets created
- No IGW created
- No DHCP options set created
- No DNS resolution created

Deleting Your Default Subnets and Default VPC

- You can delete a default subnet or default VPC just as you can delete any other subnet or VPC but you must create a new VPC or subnet first. If you delete your default subnets or default VPC, you must explicitly specify a subnet in another VPC in which to launch your instance, because you can't launch instances into EC2. If you do not have another VPC, you must create a non-default VPC and non-default subnet.
- If your VPC subnet has instances launched (EC2, ..) you will not be able to delete it until you terminate these instances

- In case you have Elastic Network Interfaces (ENIs) on the subnet that would not terminate with the instances, you have to delete or re-associate these away from the subnet

Subnets

- A subnet only can be associated with a single availability zone
- Subnet can be configured with an Internet gateway to enable communication over the Internet, or virtual private gateway (VPN) connection to enable communication with your corporate network
- For Subnets not connected to the Internet, but has traffic routed through Virtual Private Gateway only is termed as VPN-only subnet
- Subnets can be configured to Enable assignment of the Public IP address to all the Instances launched within the Subnet by default, which can be overridden during the creation of the Instance
- Subnet Sizing
  - If you create one subnet equal in size to the vpc CIDR block, you will not be able to create any other subnet in the VPC because any other one that belongs to the CIDR block will overlap with the first one
    - **Solution:** Delete the one you created and then carefully design and deploy your IP address scheme
  - CIDR block assigned to the Subnet can be a subset of the VPC CIDR, which allows you to launch multiple subnets within the VPC
  - You can not create overlapping IP address subnets in a VPC
- Subnet Routing
  - Each Subnet is associated with a route table which controls the traffic.
  - The route table by default has a rule with connect the subnets between Azs
  - Default route table can be deleted if we create a new one and we mark it as the default one, then, we can delete the one create automatically.
- Subnet Security
  - Subnet security can be configured using Security groups and NACLs
  - Security groups works at instance level, NACLs work at the subnet level
  -

• Key points to remember

- **VPC (Virtual Private Cloud) is a virtual network dedicated to a customer on AWS.**
- VPC cannot be associated with multiple regions.
- VPC or Subnets IP address range cannot be modified once the VPC has been created.
- A newly VPC created has the following resources:
  - Subnets.
  - Route Tables.
  - Dynamic Host Configuration Protocol (DHCP).
  - Security Groups.
  - Network Access Control
- A subnet is segmentation of a VPC.
- The smallest subnet can have 16 IPs (/28 netmask) IP addresses and maximum 65,536 IPs (/16 netmask).

- 5 IP addresses of a subnet is being used by AWS, so if you create a subnet with 16 IP addresses will have only 11 IP address available to use (16-5 = 11)
- **There are three types of a subnet:**
    - **Public:** Has internet access and traffic is routed to the IGW (Internet Gateway)
    - **Private**: Is not routed to the IGW. (No internet access)
    - **VPN-only Subnet**: Private subnet traffic is directed to a VPG (Virtual Private Gateway) using a VPN connection to your on-premise Corporate Network.

IP Addresses

Instances launched in the VPC can have Private, Public and Elastic IP address assigned to it and are properties of ENI (Network Interfaces)

- **Private IP Addresses**
    - Private IP addresses are not reachable over the Internet, and can be used for communication only between the instances within the VPC
    - Primary IP address is associated with the network interface for its lifetime, even when the instance is stopped and restarted and is released only when the instance is terminated
    - Additional Private IP addresses, known as secondary private IP address, can be assigned to the instances and these can be reassigned from one network interface to another
- **Public IP address**
    - Public IP addresses are reachable over the Internet, and can be used for communication between instances and the Internet, or with other AWS services that have public endpoints
    - Public IP address assignment to the Instance depends if the Public IP Addressing i**s enabled for the Subnet.**
    - Public IP address can also be assigned to the Instance by enabling the Public IP addressing during the creation of the instance, which overrides the subnet's public IP addressing attribute
    - Public IP address is assigned from AWS pool of IP addresses and it is not associated with the AWS account and hence is released when the instance is stopped and restarted or terminated.
- **Elastic IP address**
    - Elastic IP addresses are static, persistent public IP addresses which can be associated and disassociated with the instance, as required
    - Elastic IP address is allocated at an VPC and owned by the account unless released
    - A Network Interface can be assigned either a Public IP or an Elastic IP. If you assign an instance, already having an Public IP, an Elastic IP, the public IP is released
    - Elastic IP addresses can be moved from one instance to another, which can be within the same or different VPC within the same account
    - Elastic IP are charged for non usage i.e. if it is not associated or associated with a stopped instance or an unattached Network Interface
    - 5 (EIP) per region (Your need to fill a form to ask for more EIPs

- **An Elastic IP has to be on the IGW and cannot be directly attached to the instance**

Elastic Network Interface (ENI)

- You can create and attach an additional network interface, known as an elastic network interface (ENI) to any instance in your VPC. The number of ENIs you can attach varies by instance type
- Each Instance is attached with default elastic network interface (Primary Network Interface eth0) and cannot be detached from the instance
- ENI can include the following attributes
  - Primary private IP address
  - One or more secondary private IP addresses
  - One Elastic IP address per private IP address
  - One public IP address, which can be auto-assigned to the network interface for eth0 when you launch an instance, but only when you create a network interface for eth0 instead of using an existing ENI
  - One or more security groups
  - A MAC address
  - A source/destination check flag
  - A description
- ENI's attributes follow the ENI as it is attached or detached from an instance and reattached to another instance. When an ENI is moved from one instance to another, network traffic is redirected to the new instance.
- Multiple ENIs can be attached to an instance and is useful for use cases:
  - Create a management network.
  - Use network and security appliances in your VPC.
  - Create dual-homed instances with workloads/roles on distinct subnets.
  - Create a low-budget, high-availability solution.

Route Tables

- Route table defines rules, termed as routes, which determine where network traffic from the subnet would be routed
- Each VPC has a implicit router to route network traffic
- Each VPC has a Main Route table, and can have multiple custom route tables created
- Each Subnet within a VPC must be associated with a single route table at a time, while a route table can have multiple subnets associated with it
- Subnet, if not explicitly associated to a route table, is implicitly associated with the main route table
- Every route table contains a local route that enables communication within a VPC which cannot be modified or deleted
- Route priority is decided by matching the most specific route in the route table that matches the traffic
- Route tables needs to be updated to defined routes for Internet gateways, Virtual Private gateways, VPC Peering, VPC Endpoints, NAT Device etc.

Internet Gateways – IGW

- An Internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in the VPC and the Internet.
- IGW imposes no availability risks or bandwidth constraints on the network traffic.
- An Internet gateway serves two purposes:
  - To provide a target in the VPC route tables for Internet-routable traffic,
  - To perform network address translation (NAT) for instances that have been assigned public IP addresses.
- Enabling Internet access to an Instance requires
  - Attaching Internet gateway to the VPC
  - Subnet should have route tables associated with the route pointing to the Internet gateway
  - Instances should have a Public IP or Elastic IP address assigned
  - Security groups and NACLs associated with the Instance should allow relevant traffic

Deleting an Internet Gateway

You can not delete an IGW that is attached to a VPC. **(!) You have to detach the IGW from the VPC first, then you can delete it**
Likewise, you can not delete a VPC that has an IGW attached to it, you have to detach the IGW first, then you can delete the VPC (if it had no EC2 instances connected to it)

Security Groups

- An EC2 instance, when launched, can be associated with one or more security groups with the instance, which acts as a virtual firewall that controls the traffic to that instance
- Security groups have specify rules that control the inbound traffic that's allowed to reach the instances and the outbound traffic that's allowed to leave the instance
- Security groups are associated with network interfaces and they operate at the network interface level of EC2 instance. Changing an instance's security groups changes the security groups associated with the primary network interface (eth0)
- An EC2 instance's Elastic Network Interface (ENI) can have up to 5 security groups and with 50 rules per security group
- Rules for a security group can be modified at any time; the new rules are automatically applied to all instances associated with the security group.
- All the rules from all associated security groups are evaluated to decide where to allow traffic to an instance
- **Security Group features**
  - **Default security group:**
    - Inbound: It allows all inbound traffic from other instances associated with the default security group.

- All outbound traffic is allowed by default. Default security group example:

Inbound				
Type	**Protocol**	**Port Range**	**Source**	**Description**
All traffic	**All**	**All**	**sg-d97ab (default)**	
Outbound				
All traffic	**All**	**All**	**0.0.0.0/0**	

  - **Custom (created by hand) - Security group:**
    - No inbound rules - Basically all inbound traffic is denied by default
    - All outbound traffic is allowed by default
  - Security groups are stateful — Stateful means: What is allowed to IN it is allowed to OUT, even there is not any rule for the outbound traffic
  - If multiple rules for the same protocol and port the Most permissive rule is applied for e.g. for multiple rules for tcp and port 22 for specific IP and Everyone, everyone is granted access being the most permissive rule
- **To ensure traffic is immediately interrupted, use NACL as they are stateless and therefore do not allow automatic response traffic.**
- Security groups have allow only rules, **no explicit deny rules. Security group cannot have explicit deny rules because they end with an implicit deny all rule**
- A security group can span several AZs and subnets in the same VPC.
- Security groups applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later

Network ACLs

- **A Network ACLs (NACLs) act as a firewall and it operates at the subnet level, controlling both inbound and outbound traffic at the subnet level (ACLS filter traffic that is in or out the subnet)**
- ACLS are stateless — Stateless means: What is allowed to IN it is not allowed to OUT or what is allowed to OUT is not allowed to IN, and therefore, they a need a rule for INBOUND and OUTBOUND traffic. F or e.g. if you enable Inbound SSH on port 22 from the specific IP address, you would need to add a Outbound rule for the response as well
- **It can have allow and deny rules.**
- Network ACL has separate inbound and outbound rules
- Default/Custom ACLs:
  - **Default ACL:** Allows all inbound and outbound traffic.
  - **Custom ACL**: Denies all inbound and outbound traffic until we add rules.
- Network ACL is a numbered list of rules that are evaluated in order
- **To deny traffic from an Network IPs range, you must do it with ACLs not with Security Groups**
- ACLS automatically applies to all instances in the subnet associated with the ACL (backup layer of defense, so you don't have to wait/rely on someone specifying the security group)

- **ACLS, rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it's applied immediately regardless of any higher-numbered rule that may contradict it. (SAME as Iptables in Linux)**

Network Address Translation – (NAT)

- **Network Address Translation (NAT) devices, launched in the public subnet, enables instances in a private subnet to connect to the Internet, but prevents the Internet from initiating connections with the instances.**
- Instances in private subnets would need internet connection for performing software updates or trying to access external services
- NAT device performs the function of both address translation and port address translation (PAT)
- NAT instance prevents instances to be directly exposed to the Internet and having to be launched in Public subnet and assignment of the Elastic IP address to all, which are limited.
- NAT device routes the traffic, from the private subnet to the Internet, by replacing the source IP address with its address and for the response traffic it translates the address back to the instances' private IP addresses.
- **AWS allows NAT configuration in 2 ways**
  - **NAT Instance**
  - **NAT Gateway, managed service by AWS**

NAT device Configuration Key Points

- It needs to be launched in the Public Subnet
- It needs to be associated with an Elastic IP address (or public IP address)
- It should have the Source/Destination flag disabled to route traffic from the instances in the private subnet to the Internet and send the response back
- It should have a Security group associated that:
    - Allows Outbound Internet traffic from instances in the private subnet
    - Disallows Inbound Internet traffic from everywhere
- Instances in the private subnet should have the Route table configured to direct all Internet traffic to the NAT device

NAT Gateway

- **NAT gateway is an AWS managed NAT service (AWS provides security/patching for the NAT gateway) that provides better availability, higher bandwidth, and requires less administrative effort. It scales automatically.**
- A NAT gateway supports bursts of up to 10 Gbps of bandwidth.
- For over 10 Gbps bursts requirement, the workload can be distributed by splitting the resources into multiple subnets, and creating a NAT gateway in each subnet
- NAT gateway is associated with One Elastic IP address which cannot be disassociated after it’s creation. It cannot be associated with a Public IP only EIP.
- Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone.
- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- NAT gateway cannot be associated a security group. Security can be configured for the instances in the private subnets to control the traffic
- Network ACL can be used to control the traffic to and from the subnet. NACL applies to the NAT gateway’s traffic, which uses ports 1024-65535
- NAT gateway when created receives an elastic network interface that’s automatically assigned a private IP address from the IP address range of the subnet. Attributes of this network interface cannot be modified
- NAT gateway cannot send traffic over VPC endpoints, VPN connections, AWS Direct Connect, or VPC peering connections. Private subnet’s route table should be modified to route the traffic directly to these devices.

NAT Instance

- **NAT Instances are created/managed by the customer (you are responsible of the security/patching of the NAT instance).**
- NAT Instance it has to be always on the public subnet because it needs access to internet and it can be used with a public IP address or an Elastic IP address

- NAT instance can be created by using Amazon Linux AMIs configured to route traffic to Internet.
- They do not provide the same availability and bandwidth and need to configured as per the application needs.
- NAT instances must have security groups associated with Inbound traffic enabled from private subnets and Outbound traffic enabled to the Internet
- NAT instances should have the Source Destination Check attribute disabled, as it is neither the source nor the destination for the traffic and merely acts as a gateway

NAT Instance as a Proxy

- NAT instance does a PAT function, so it can hide more than one device/ EC2 instance behind it when the traffic is initiated from these instances behind the NAT instance (so if you need to connect several EC2 in two AZs through two fixed IPs to your company you need to connect the EC2 to two NAT instances with one Elastic IP each one and the NAT instance acts as Proxy)

High Availability NAT Instance

- Create One NAT instance per Availability Zone. NAT instances can be deployed across AZs independently for HA. EC2 instances, in each AZ, on private subnets requiring NAT should be configured to use the local NAT instance in the respective AZ
- Configure all Private subnet route tables to the same zone NAT instance
- Use Auto Scaling for NAT availability
- Use Auto Scaling group per NAT instance with min and max size set of 1. So if NAT instances fail, Auto Scaling will automatically launch an replacement instance
- NAT instance is highly available with limited downtime
- Let Auto Scaling monitor health and availability of the NAT instance
- Bootstrap scripts with the NAT instance to update the Route tables programmatically
- Keep a close watch on the Network Metrics and scale vertically the NAT instance type to the one with high network performance

Disabling Source/Destination checks

- By default, any EC2 instance will not allow traffic except that it is the source of destination of itself and you can disable the source/destination check for an EC2 instance when it is running or when it is stopped from the AWS console or using a CLI
- This is done by the attribute **"source/destination check" of the EC2 instance" . It is enabled by default, this is why the EC2 instance, when the attribute is enabled, will not allow traffic through it**
- Each EC2 instance performs source/destination checks, by default, and the instance must be the source or destination of any traffic it sends or receives

- However, as the NAT instance acts as a router between the Internet and the instances in the private subnet it must be able to send and receive traffic when the source or destination is not itself.
- **Therefore, the source/destination checks on the NAT instance should be disabled**

Key points to remember In the exam:

- If there is a server configured in a public subnet, obviously, it does not need any NAT service configured because it already has an Internet Gateway configured and Internet access (it looks a stupid, but in the exam, maybe you don't pay attention to this)
- If there is a DB server in a private subnet and it needs to get updates/patching, it means it needs Internet access and therefore, we need a rule on the NAT Instance to allow source from the Public subnet to the NAT Instance direction Inbound Port 80 or HTTPS 443 or by SFTP (over SSH 22)/FTP (21) and from the NAT Instance direction Outbound to 0.0.0.0/0 Port 80 in order to receive updates and you should disable the source/destination check attribute on the NAT instance in order to the trafic can go and return through the NAT Instance. Also, you need to make sure the private subnet route table points at the NAT instance ID for traffic destined to the internet (0.0.0.0/0). Lastly, make sure the security group of the private subnet instances and that of the NAT instance and the respective N ACLS are configured properly

- A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them privately. 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, with a VPC in another AWS account, or with a VPC in a different AWS Region.

VPC peering connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between then using private IP addresses.

- **VPC peering connection:**
  - Lets you connect instances from other VPC's together. Instances in either VPC can communicate with each other as if they are within the same network
  - Connections are initiated through a request/accept protocol.
  - It was not possible to create VPC peering connection between VPCs in different regions. (NOTE – VPC Peering is now supported inter-region.) VPCs between regions is possible since 2017 and the connection is encrypted after having create the VPC peering. It is necessary to edit the route table from each of the the implied

regions or implied accounts indicating as the origin the local network from each VPC and the VPC peering ID as target

- VPC peering connection can be established between your own VPCs, or with a VPC in another AWS account in a single different region.
- It is a service provided by AWS and therefore fault tolerant and HA. There is no single point of failure for communication or a bandwidth bottleneck
- AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware.
- **VPC Peering Rules & Limitations**
    - VPC peering connection cannot be created between VPCs that have matching or overlapping CIDR blocks, therefore, it cannot be defined the same IP/CIDR block in the two VPCs.
    - VPC peering connection are limited on the number active and pending VPC peering connections that you can have per VPC.VPC Peering Connections Limited to 50 vpcs connections.
    - Maximum Transmission Unit (MTU) across a VPC peering connection is 1500 bytes.
    - A placement group can span peered VPCs that are in the same region; however, you do not get full-bisection bandwidth between instances in peered VPCs
    - VPC peering does not support transitive peering relationships/ routing. In a VPC peering connection, the VPC does not have access to any other VPCs that the peer VPC may be peered with even if established entirely within your own AWS account. **VPC peering connection is a one to one relationship between two VPCs.** You can create multiple VPC peering connections for each VPC that you own but Only one VPC peering connection can be established between the same two VPCs at the same time
        - If you have 3 VPCs you can peered three VPCs together in a full mesh (malla completa) configuration:
            - VPC A peeered to VPC B through VPC peering connection pcx-1
            - VPC A peeered to VPC C through VPC peering connection pcx-2
            - VPC B peeered to VPC C through VPC peering connection pcx-3
                - (Like performing a triangle)
            - You need to edit the route tables for each VPC point to the relevant VPC peering connection to access the entire CIDR block of the peer VPCs, You may want to use this full mesh configuration when you have separate VPCs that need to share resources with each other without restriction, for example, as a file sharing system
    - **VPC peering does not support Edge to Edge Routing Through a Gateway or Private Connection**. It is only possible to connect a VPC with another VPC but a VPC can be

connected with several VPCS although the rest of VPCs dont see between each other. For example:
    - In the figure VPC-A and VPC-B are peered, VPC-A is also connected to the Corporate network by a VPN or Direct connect connection
    - Corporate network can NOT access VPC-B by being connected to VPC-A, because VPC peering will not allow VPC-B routes to be advertised to Corporate network and vice versa
    - Edge to edge routing here refers to routing external traffic (to the peered VPCs) through the VPC peering connection
- **In a VPC peering connection, the VPC does not have access to any other connection that the peer VPC may have and vice versa. Connections that the peer VPC can include:**
    - A VPN connection or an AWS Direct Connect connection to a corporate network
    - An Internet connection through an Internet gateway
    - An Internet connection in a private subnet through a NAT device
    - A VPC endpoint to an AWS service; for example, an endpoint to S3.

  - To enable traffic, it is neccesary to add the routes to the route's table and create the security groups.
  - Any tags created for the VPC peering connection are only applied in the account or region in which they were created
  - Unicast reverse path forwarding in VPC peering connections is not supported
  - Instance's public DNS hostname does not resolve to its private IP address across peered VPCs.
- **VPC Peering Architecture**
  - VPC Peering can be applied to create shared services or perform authentication with an on-premises instance
  - This would help creating a single point of contact, as well limiting the VPN connections to a single account or VPC

AWS VPC VPN – CloudHub Connections

VPC VPN (Virtual Private Network) Connections

- Virtual Private Gateway (VGW) is the gateway the main table refers to access to customer premise/data center
- VPC VPN connections are used to extend on-premise data centers to AWS
- VPC VPN connections provide secure IPSec connections from on-premise computers/services to AWS
- You can have up to 10 IPsec connections per VGW (A IPsec connections means a VPN connection from a site/location) (soft limit can be increased by contacting AWS)

- **VPN types:**
  - **AWS hardware VPN**
    - Connectivity can be established by creating an IPSec, hardware VPN connection between the VPC and the remote network.
    - On the AWS side of the VPN connection, a Virtual Private Gateway (VGW) provides two VPN endpoints for automatic fail-over.
    - On customer side a customer gateway (CGW) needs to be configured, which is the physical device or software application on the remote side of the VPN connection
  - **AWS Direct Connect connection**
    - AWS Direct Connect provides a dedicated private connection from a remote network to your VPC.
    - Direct Connect can be combined with an AWS hardware VPN connection to create an IPsec-encrypted connection
  - **AWS VPN CloudHub**
    - For more than one remote network for e.g. multiple branch offices, multiple AWS hardware VPN connections can be created via the VPC to enable communication between these networks
  - **Software VPN**
    - VPN connection can be setup by running a software VPN like OpenVPN appliance on an EC2 instance in the VPC
    - AWS does not provide or maintain software VPN appliances; however, there are range of products provided by partners and open source communities

VPN Connection VPN Components

- **Virtual Private Gateway – VGW**
  - A virtual private gateway is the VPN concentrator on the AWS side of the VPN connection
- **Customer Gateway – CGW**
  - A customer gateway is a physical device or software application on customer side of the VPN connection.
  - When a VPN connection is created, the VPN tunnel comes up when traffic is generated from the remote side of the VPN connection.
  - VGW is not the initiator; CGW must initiate the tunnels
  - If the VPN connection experiences a period of idle time, usually 10 seconds, depending on the configuration, the tunnel may go down. To prevent this, a network monitoring tool to generate keep-alive pings; for e.g. by using IP SLA.
  -

VPN Configuration

- VPC has an attached Virtual Private Gateway (VGW), and the remote network includes a customer gateway, which must be configured to enable the VPN connection.
- Routing must be set-up so that any traffic from the VPC bound for the remote network is routed to the Virtual Private Gateway (VGW).
- Each VPN has two tunnels associated with it that can be configured on the customer router, as is not single point of failure
- Multiple VPN connections to a single VPC can be created, and a second CGW can be configured to create a redundant connection to the same

external location or to create VPN connections to multiple geographic locations.

VPN Routing Options

- For a VPN connection, the route table for the subnets should be updated with the type of routing (static of dynamic) that you plan to use.
- Route tables determine where network traffic is directed. Traffic destined for the VPN connections must be routed to the Virtual Private Gateway (VGW).
- Type of routing can depend on the make and model of your VPN devices.
- Static Routing
  - If your device does not support BGP, specify static routing.
  - Using static routing, the routes (IP prefixes) can be specified that should be communicated to the Virtual Private Gateway (VGW).
  - Devices that don’t support BGP may also perform health checks to assist fail-over to the second tunnel when needed.
- BGP dynamic routing
  - If the VPN device supports Border Gateway Protocol (BGP), specify dynamic routing with the VPN connection.
  - When using a BGP device, static routes need not be specified to the VPN connection because the device uses BGP for auto discovery and to advertise its routes to the Virtual Private Gateway (VGW).
  - BGP-capable devices are recommended as the BGP protocol offers robust liveness detection checks that can assist fail-over to the second VPN tunnel if the first tunnel goes down.
- Only IP prefixes known to the Virtual Private Gateway (VGW), either through BGP advertisement or static route entry, can receive traffic from your VPC.
- Virtual Private Gateway (VGW) does not route any other traffic destined outside of the advertised BGP, static route entries, or its attached VPC CIDR.

VPN Connection Redundancy

- A VPN connection is used to connect the customer network to a VPC.
- Each VPN connection has two tunnels to help ensure connectivity in case one of the VPN connections becomes unavailable, with each tunnel using a unique Virtual Private Gateway (VGW) public IP address.
- Both tunnels should be configured for redundancy.
- When one tunnel becomes unavailable, for e.g. down for maintenance, network traffic is automatically routed to the available tunnel for that specific VPN connection.
- To protect against a loss of connectivity in case the Customer Gateway (CGW) becomes unavailable, a second VPN connection can be setup to the VPC and Virtual Private Gateway (VGW) by using a second customer gateway.
- Customer gateway IP address for the second VPN connection must be publicly accessible.

- By using redundant VPN connections and CGWs, maintenance on one of the customer gateways can be performed while traffic continues to flow over the second customer gateway's VPN connection.
- Dynamically routed VPN connections using the Border Gateway Protocol (BGP) are recommended, if available, to exchange routing information between the customer gateways and the virtual private gateways.
- Statically routed VPN connections require static routes for the network to be entered on the customer gateway side.
- BGP-advertised and statically entered route information allow gateways on both sides to determine which tunnels are available and reroute traffic if a failure occurs.

VPN CloudHub

- **VPN CloudHub can be used to provide secure communication between sites, for example, if you want to connect multiple locations to the AWS then you are configuring a VPN CloudHub**
- Spokes can communicate with each other and with the VPC (each VPN connection can talk with the other VPNs connection location so you can see the other branches for your company from that side not from AWS side)
- You pay for any traffic that goes out from the Virtual Private Gateway (VGW)
- VPN CloudHub operates on a simple hub-and-spoke model that can be used with or without a VPC.
- Design is suitable for customers with multiple branch offices and existing
- Internet connections who'd like to implement a convenient, potentially low-cost hub-and-spoke model for primary or backup connectivity between these remote offices
- AWS VPN CloudHub requires a Virtual Private Gateway (VGW) with multiple Customer Gateways. (CGW).
- Each customer gateway must use a unique Border Gateway Protocol (BGP) Autonomous System Number (ASN)
- Customer gateways advertise the appropriate routes (BGP prefixes) over their VPN connections.
- Routing advertisements are received and re-advertised to each BGP peer, enabling each site to send data to and receive data from the other sites.
- Routes for each spoke must have unique ASNs and the sites must not have overlapping IP ranges.
- Each site can also send and receive data from the VPC as if they were using a standard VPN connection.
- Sites that use AWS Direct Connect connections to the Virtual Private Gateway (VGW) can also be part of the AWS VPN CloudHub.
- To configure the AWS VPN CloudHub,
  - Multiple customer gateways can be created, each with the unique public IP address of the gateway and the ASN.
  - A VPN connection can be created from each customer gateway to a common Virtual Private Gateway (VGW).

- Each VPN connection must advertise its specific BGP routes. This is done using the network statements in the VPN configuration files for the VPN connection.

VPN-only Subnet

A public vpn-only subnets created by the VPC Wizard refers to the fact that no NAT instance (o gateway) will be created as part of the wizard configuration and therefore it is not created neither a public subnet

- **vpn-only means a subnet that does not require access to the internet**
- VPCs main route table is associated with the vpn-only subnet
- It will create a VGW but without an Elastic IP
- It will create a VPC connection
- VPN keys
- Virtual Private Gateway (VGW) is the gateway the main table refers to access to customer premise/data Center and it is used to connect the AWS Private Subnet with the Corporate Network (Data Center)
- **It is cost effective. If we are in a hurry to connect our on-premises services with AWS because it goes over Internet. VPN provides for a quick, low cost, connectivity solution (Exam tip)**
- Virtual Private Network Provide secure communication between sites
- VPN CloudHub can be used to provide secure communication between sites, for example, if you want to connect multiple locations to the AWS then you are configuring a VPN CloudHub
- You need a Customer Gateway (CGW) on your Corporate Network side with a Public IP and a Virtual Private Gateway fon the AWS side.
- **If there is a current Direct Connect Connection from a Company to AWS and you need to set up a more fault tolerant in a cost effective way, you can set up a VPN between company and AWS VPC and have the VPN and the Direct Connect Connection at the same time from the company (Exam tip)**
- Virtual Private Gateway (VGW) supports IPSec connection, it support 10 IPSec locations by default. If you need more, you need to apply for it to AWS.

AWS Direct Connect Connection (DX)

- **AWS Direct Connect Connection** is a direct connection (not internet based) and provides for Highest Speed (bandwith), less latency and higher performance than internet and it provides an alternative to using the Internet to utilize AWS cloud services
- You can NOT use a NAT instance/gateway in your VPC over the Direct Connect Connection
- Direct connect takes time to deploy, but provides guaranteed, dedicated, low latency performance
- AWS takes care of the redundancy and HA between VPC-A and VPC-B when peered together

- AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard 1 Gbps or 10 Gbps Ethernet fibre-optic cable with one end of the cable connected to your router, the other to an AWS Direct Connect router.
- Direct Connect connection can be established with 1Gbps and 10Gbps ports. Speeds of 50Mbps, 100Mbps, 200Mbps, 300Mbps, 400Mbps, and 500Mbps can be ordered from any APN partners supporting AWS Direct Connect.
- AWS Direct Connect helps to create virtual interfaces directly to the AWS cloud for e.g, to EC2 & S3 and to Virtual Private Cloud (VPC), bypassing Internet service providers in the network path.
- **AWS Direct Connect location usa VLANS (virtual interface VIF 802.1q) to map/connect a Customer Router Data Center/Location to the AWS Direct Connect Router. A Virtual interface (VIF) is basically a 802.1Q VLAN mapped from the customer router to the Direct Connect router**
    - **You need one Private VIF to connect to your private VPC subnets and one Public VIF to connect your AWS public services**
        - **One VIF to connect to your private VPC subnets**
        - **One VIF to connect to your AWS public service**
- It is not possible to use a NAT service in the VPC over
- Each AWS Direct Connect location enables connectivity to all Availability Zones within the geographically nearest AWS region.

Direct Connect Anatomy

- Amazon maintains AWS Direct Connect PoP across different locations (referred to as Colocation Facilities) which are different from AWS regions
- As a consumer, you can either purchase a rack space or use any of the AWS APN Partner which already have the infrastructure within the Colocation Facility and configure a Customer Gateway
- Connection between the AWS Direct Connect PoP and the Customer gateway within the Colocation Facility is called Cross Connect.
- Connection from the Customer Gateway to the Customer Data Center can be establish using any Service Provider Network
- Once a Direct Connect connection is created with AWS, a LOA-CFA (Letter Of Authority – Connecting Facility Assignment) would be received.
- LOA-CFA can be handover to the Colocation Facility or the APN Partner to establish the Cross Connect
- Once the Cross Connect and the connectivity between the CGW and Customer DataCenter is established, Virtual Interfaces can be created
- AWS Direct Connect requires a VGW to access the AWS VPC
- Virtual Interfaces
    - Each AWS Direct Connect connection requires a Virtual Interface
    - Each AWS Direct Connect connection can be configured with one or more virtual interfaces.
    - Public Virtual Interface can be created to connect to public resources for e.g. SQS, S3, EC2, Glacier etc which are reachable publicly only.
    - Private virtual interface can be created to connect to the VPC for e.g. instances with private ip address

    - Each virtual interface needs a VLAN ID, interface IP address, ASN, and BGP key.
- To use your AWS Direct Connect connection with another AWS account, you can create a hosted virtual interface for that account. These hosted virtual interfaces work the same as standard virtual interfaces and can connect to public resources or a VPC.

Direct Connect Advantages

- **Reduced Bandwidth Costs**
    - All data transferred over the dedicated connection is charged at the reduced AWS Direct Connect data transfer rate rather than Internet data transfer rates.
    - Transferring data to and from AWS directly reduces your bandwidth commitment to your Internet service provider
- **Consistent Network Performance**
    - Direct Connect provides a dedicated connection and a more consistent network performance experience as compared to the Internet which can widely vary
- **AWS Services Compatibility**
    - Direct Connect is a network service and works with all of the AWS services like S3, EC2 and VPC
- **Private Connectivity to AWS VPC**
    - Using Direct Connect Private Virtual Interface a private, dedicated, high bandwidth network connection can be established between your network and VPC
- **Elastic**
    - Direct Connect can be easily scaled to meet the needs by either using a higher bandwidth connection or by establishing multiple connections.

Direct Connect Redundancy

Direct Connect connections do not provide redundancy and have multiple single point of failures to the hardware devices as each connection consists of a single dedicated connection between ports on your router and an Amazon router

Redundancy can be provided by:

- Establishing a second Direct Connect connection, preferably in a different Colocation Facility using different router and AWS Direct Connect PoP
- IPsec VPN connection between the Customer DC to the VGW

Direct Connect vs IPSec VPN Connections

- A VPC VPN Connection utilizes IPSec to establish encrypted network connectivity between your intranet and Amazon VPC over the Internet.
- VPN Connections can be configured in minutes and are a good solution for immediate needs, have low to modest bandwidth requirements, and can tolerate the inherent variability in Internet-based connectivity.
- AWS Direct Connect does not involve the Internet; instead, it uses dedicated, private network connections between your intranet and Amazon VPC

- VPN connections are very cheap ($37.20/month as of now) as compared to Direct Connect connection as it requires actual hardware and infrastructure and might go in thousands.

VPN / Direct Connect - Keys notes for the exam

- **When it is time sensitive and critical data, then use a Direct Connect connection instead of Virtual Private Gateway. If in the exam, they say the connection needs to be also Fault Tolerant, it means it must have two Direct Connection using two Customer routers and two VIFS to two different Direct Connect Routers.**

• VPC Endpoints

- **VPC Endpoints lets you create an end-to-end connection from your VPC with AWS services such as S3 via a private network without the need of an internet connection, NAT Gateway etc. A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.**

- **VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network. For example, to access to sensitive information from an EC2 to S3, you can access through a VPC endpoint for Amazon S3.**

- Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.
- Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.
- There are two types of VPC endpoints:
  - Interface
    - An interface endpoint is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported service. The following services are supported:
      - **Amazon CloudWatch Logs**
      - **AWS CodeBuild**
      - **Amazon EC2 API**
      - **Elastic Load Balancing API**
      - **AWS Key Management Service**
      - **Amazon Kinesis Data Streams**
      - **AWS Service Catalog**
      - **Amazon SNS**

        - **AWS Systems Manager**
        - **Endpoint services hosted by other AWS accounts**
        - **Supported AWS Marketplace partner services**

    - Gateway.
        - A gateway endpoint is a gateway that is a target for a specified route in your route table, used for traffic destined to a supported AWS service. The following AWS services are supported:
            - Amazon S3
            - DynamoDB
- VPC endpoint enables creation of a private connection between your VPC and another AWS service using its private IP address
- Endpoints currently do not support cross-region requests, ensure that the endpoint is created in the same region as your bucket
- AWS currently supports endpoints for S3 service and DynamoDB (it is also supported since 2017)

Configuration

- Endpoint requires the VPC and the service to be accessed via the endpoint
- Endpoint needs to be associated with the Route table and the route table cannot be modified to remove the route entry. It can only be deleted by removing the Endpoint association with the Route table
- A route is automatically added to the Route table with a destination that specifies the prefix list of service and the target with the endpoint id. for e.g. A rule with destination pl-68a54001 (com.amazonaws.us-west-2.s3) and a target with this endpoints' ID (e.g. vpce-12345678) will be added to the route tables
- Access to the resources in other services can be controlled by endpoint policies
- Security groups needs to be modified to allow Outbound traffic from the VPC to the service that's specified in the endpoint. Use the service prefix list ID for e.g. com.amazonaws.us-east-1.s3 as the destination in the outbound rule
- Multiple endpoint routes to different services can be specified in a route table, and multiple endpoint routes to the same service can be specified in different route tables, but you cannot have multiple endpoints to the same service in a single route table

Limitations

- Endpoint cannot be tagged
- Endpoint cannot be transferred from one VPC to another, or from one service to another
- Endpoint connections cannot be extended out of a VPC i.e. resources across the VPN connection, VPC peering connection, AWS Direct Connect connection cannot use the endpoint

Controlling the Use of VPC Endpoints

By default, IAM users do not have permission to work with endpoints. You can create an IAM user policy that grants users the permissions to create, modify, describe, and delete endpoints.

EC2

EC2 features

- Virtual computing environments, known as Ec2 instances
- Preconfigured images templates for your instances, known as Amazon Machine Images (AMIs).
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as Instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as **Instance store volumes**
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as **Amazon EBS volumes**
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Metadata, known as tags, can be created and assigned to EC2 resources

Accessing EC2

- Amazon EC2 console
  - Amazon EC2 console is the web-based user interface which can be accessed from AWS management console
- AWS Command line Interface (CLI)
  - Provides commands for a broad set of AWS products, and is supported on Windows, Mac, and Linux.
- AWS Tools for Windows Powershell
  - Provides commands for a broad set of AWS products for those who script in the PowerShell environment
- AWS Query API
  - Query API allows for requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named Action
- AWS SDK libraries
  - AWS provide libraries in various languages which provide basic functions that automate tasks such as cryptographically signing your requests, retrying requests, and handling error responses

EC2 - Amazon Machine Image – AMI

An Amazon Machine Image (AMI) is basically an template and can be used to launch as many instances as needed

An AMI includes the following:

- A template for the root volume for the instance for e.g. an operating system, an application server, and applications
- Launch permissions that control which AWS accounts can use the AMI to launch instances for e.g. AWS account ids with whom the AMI is shared
- A block device mapping that specifies the volumes to attach to the instance when it's launched

AMIs can be either

- AWS managed, provided and published AMIs
- Third party or Community provided public custom AMIs
- Private AMIs created by other AWS accounts and shared with you
- Private and Custom AMIs created by you

AMI lifecycle

- After you create and register an AMI tou can use it to launch new instances. (You can also launch instances from an AMI if the AMI owner grants you launch permissions.)
- You can copy an AMI to the same region or to different regions.
- When you are finished launching instance from an AMI, you can deregister the AMI

AMI Types

- **Region & Availability Zone**
  - AMIs are specific to a region and if needed in other region must be copied over
- **Operating system**
  - linux, windows, ubuntu etc
  - Architecture (32-bit or 64-bit)
- **Launch Permissions**
  - Launch permissions define who has access to the AMI
    - Public – Accessible to all AWS accounts
    - Explicit – Shared with specific AWS accounts
    - Private – Owned and available for AMI creator AWS account only
- **Root device storage**
  - AMIs can have EBS or Instance store as the root device storage
  - **EBS backed**
    - EBS volume are independent of the EC2 instance lifecycle and can persist independently
    - EBS backed instances can be stopped without losing the volumes
    - EBS instance can also be persist without losing the volumes on instance termination, if the Delete On Termination flag is disabled
    - EBS backed instances boot up uch faster than the Instance store backed instances as only the parts required to boot the instance

needs to be retrieved from the snapshot before the instance is made available
        - AMI creation is much easier for AMIs backed by Amazon EBS. The CreateImage API action creates your Amazon EBS-backed AMI and registers it
    - **Instance Store backed**
        - Instance store is an ephemeral storage and is dependent on the lifecycle of the Instance
        - Instance store is deleted if the instance is terminated or if the EBS backed instance, with attached instance store volumes, is stopped
        - Instance store volumes cannot be stopped
        - Instance store volumes have their AMI in S3 and have higher boot times compared to EBS backed instances, as all the parts have to be retrieved from Amazon S3 before the instance is made available
        - To create Linux AMIs backed by instance store, you must create an AMI from your instance on the instance itself using the Amazon EC2 AMI tools

Deregistering AMI

Charges are incurred on the AMI created and they can be deregistered, if not needed.

- Deregistering an AMI does not delete the EBS snapshots or the bundles in the S3 buckets and have to be removed separately
- Once deregistered, new instances cannot be launched from the AMI. However, it does not impact already created instances from the AMI

AMIs with Encrypted Snapshots

- AMIs, with EBS backed snapshots, can be attached with both an encrypted root and data volume
- AMIs copy image can be used to create AMIs with encrypted snapshots from AMIs with unencrypted snapshots. By default, copy image preserves the encryption status of the snapshots
- Snapshots can be encrypted with either your default AWS Key Management Service customer master key (CMK), or with a custom key that you specify

AMI Copying

- Amazon EBS-backed AMIs and instance store-backed AMIs can be copied.
- AMI copy image can be used to encrypt an AMI from an unencrypted AMI
- AMI copy image can’t be used to create an unencrypted AMI from an encrypted AMI

• EC2 Instance families

EC2 Instance types determines the hardware of the host computer used for the instance.

Instance family	Current generation instance type
General Purpose	T
Compute Optimized	More CPU than memory Compute & HPC intensive use Ex. C2, C3, C4 (C letter for computing)
Memory optimized	More RAM/memory Memory intensive apps, DB, and caching Ex. R3, R4 (R letter for RAM)
Storage Optimized	Very high, low latency, I/O I/O intensive apps, data warehousing, Hadoop Ex. I2, D2 (I letter for intensive and D letter for Database)
GPU compute instances	Graphics optimized High performance and parallel computing Ex. G2 (G letter for GPU)

T2 Instances (General Purpose) – Special case

- Mainly intended for workloads that don't use the full CPU often or consistently, but occasionally need to burst.
- Are available as On-Demand or Reserved instances, but do not allow spot instances
- By default, 20 (soft limit) T2 instances can run simultaneously
- Cannot be launched as a Dedicated instance
- T2 CPU Credits
  - It provides the performance of a full CPU core for one minute
  - T2 instances provide a baseline level of CPU performance, while CPU governs the ability to burst above the baseline level
  - One CPU credit is equal to one vCPU running at 100% utilization for one minute.
  - Basically, it earns CPU credits when it is using few CPU resources and it consume the CPU credits when using the CPU above the baseline.
  - After a VM STOP/START , the instance receives the initial CPU credits again
  - When credit balance is completely exhausted, the instance will perform at its baseline performance

EC2 Instance Type selection criteria

Basically, it depends on what the instance type supports: HVM or PV, EBS or Instance Store volumes, launched as EBS optimized, launched in a Placement Group, support Enhanced Networking, EBS volumes to be encrypted

• EC2 - Instance Purchasing Option

1. **Understand the difference between the three payment plans:**
    - **On-Demand instances:**
        - pay per hour
        - There might be instances during peak demand where the instance cannot be launched
        - Use cases:
            - Users that want the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment
            - Applications with short term, spiky, or unpredictable workloads that cannot be interrupted
            - Applications being developed or tested on Amazon EC2 for the first time
    - **Reserved instances (to reserve Capacity)**
        - Reserved Instances provides lower hourly running costs by providing a billing discount as well as capacity reservation that is applied to instances and there would never be a case of insufficient capacity from AWS
        - Its just an **accounting term** applied to the instance not a physical instance
        - Amazon EC2 Reserved Instances (RI) provide a significant discount (up to 75%) compared to On-Demand pricing and provide a capacity reservation when used in a specific Availability Zone.
        - Auto Scaling or other AWS services can be used to launch the On-Demand instances that use the Reserved Instance benefits
        - Reserved Instances can be modified and continue to benefit from your capacity reservation, when the computing needs change.
        - Use cases:
            - Applications with steady state or predictable usage
            - Applications that require reserved capacity
            - Users able to make upfront payments to reduce their total computing costs even further
            - Production-ready application, applications that will be live for a long time.
        - Cons
            - Reserved Instances do not renew automatically, and the EC2 instances can be continued to be used but charged On-Demand rates
            - You pay for the entire term, regardless of the usage
            - Once purchased, the reservation cannot be cancelled but can be sold in the Reserved Instance Marketplace

        - Reserved Instance pricing tier discounts only apply to purchases made from AWS, and not to the third party Reserved instances
- **Billing Benefits & Payment Options**
    - Reserved Instances are billed for every hour during the term that you select, regardless of whether the instance is running or not.
    - Reserved Instance purchase reservation is automatically applied to running instances that match the specified parameters
    - Reserved Instance can also be utilized by launching On-Demand instances with the same configuration as to the purchased reserved capacity
    - **Payment Options**
        - **No Upfront**
            - No upfront payment is required and the account is charged on a discounted hourly rate for every hour (Only available as 1-year reservation)
        - **Partial Upfront**
            - A portion of the cost is paid upfront and the remaining hours in the term are charged at an hourly discounted rate
        - **Full Upfront**
            - Full payment is made at the start of the term, with no costs for the remainder of the term.
    - Buying Reserved Instances need selection of the following
        - Platform (for example, Linux)
        - Instance type (for example, m1.small)
        - Availability Zone in which to run the instance
        - Term (time period) over which you want to reserve capacity
        - Tenancy (dedicated tenancy, as opposed to shared).
        - Offering (No Upfront, Partial Upfront, All Upfront
    - **Reserved Instance (RI) Types**
      With RIs, you can choose the type that best fits your applications needs.
        - **Standard RIs:** These provide the most significant discount (up to 75% off On-Demand) and are best suited for steady-state usage.
        - **Convertible RIs**: These provide a discount (up to 54% off On-Demand) and the capability to change the attributes of the RI as long as the exchange results in the creation of Reserved Instances of equal or greater value. Like Standard RIs, Convertible RIs are best suited for steady-state usage.
        - **Scheduled RIs:**

- Enable you to purchase capacity reservations that recur on a daily, weekly, or monthly basis, with a specified start time and duration, for a one-year term.
- Scheduled Instances are a good choice for workloads that do not run continuously, but do run on a regular schedule for e.g. weekly or monthly batch jobs

- **Spot Instances**
  - Spot instances enables bidding on unused EC2 instances, and are launched whenever the bid price exceeds the current market spot price
  - Spot instances are a cost-effective choice and can bring the EC2 costs down significantly
  - Well Suited for
    - Applications that have flexible start and end times
    - Applications that are only feasible at very low compute prices
    - Users with urgent computing needs for large amounts of additional capacity (data analysis, batch jobs, background processing, and optional tasks)
  - Spot instances differ from the On-Demand instances in that they are not launched immediately and they can be terminated any-time
  - Usual strategy involves using Spot instances with On-Demand or Reserved instances, which provide a minimum level of guaranteed compute resources, while spot instances provide an additional computation boost
  - Spot instances can also be launched with a required duration (also known as Spot blocks), which are not interrupted due to changes in the Spot price
  - **T2 does not support for Spot instances**
  - **Spot Concepts**
    - Spot pool – Pool of EC2 instances with the same instance type, availability zone, operating system and network platform
    - Spot price – Current market price of a spot instance per hour as set by Amazon EC2 based on the last fulfilled bid
    - Spot bid – is the maximum bid price the bidder is willing to pay for the spot instance
    - Spot fleet – is the set of instances launched based on the criteria the bidder
    - Spot instance interruption – EC2 terminates the spot instances whenever the bid price is lower than the current market price or the supply has reduced
    - Bid status – provides the current state of the spot bid
  - If the Spot instance is terminated by Amazon, you are not billed for the partial hour. However, if the spot instance are terminated by you, you will be charged for the partial hour

        - When EC2 marks a Spot instance for termination, it provides a Spot instance termination notice, which gives the instance a two-minute warning before it terminates.
        - Termination notice warning is made available to the applications on the Spot instance using an item in the instance metadata termination-time attribute and includes the time when the shutdown signal will be sent to the instance's operating system
        - Relevant applications on Spot Instances should poll for the termination notice at 5 second intervals, giving it almost the entire two minutes to complete any needed processing before the instance is terminated and taken back by AWS
        - Spot Instances best practices
            - Choose a reasonable bid price, which is low enough to suit your budget and high enough for the request to be fulfilled and should not be higher than the On-Demand bid price
            - Ensure the instances are up and ready as soon as the request is fulfilled, by provisioning an AMI with all the required softwares and load application data from user data
            - Store important data regularly and externally in a place that won't be affected by Spot instance termination for e.g., use S3, EBS, or DynamoDB
            - Divide the work into smaller finer tasks so that they can be completely and state saved more frequently
            - Use Spot termination notice warning to monitor instance status regularly
            - Test applications using On-Demand instances and terminating them to ensure that it handles unexpected termination gracefully
    - **There are three tenancy options:**
        - **Shared Tenancy**: Multiple instances run on the same hardware (default option)
        - **Dedicated instance:**
            - Dedicated Instances are EC2 instances that run in a VPC on hardware that's dedicated to a single customer
            - Each VPC has a related instance tenancy attribute, and can't be changed after the VPC has been created. Default is shared
                - Each instance launched into a VPC has a tenancy attribute, can't be changed after the instance is launched.
        - **Dedicated host:**
            - Your instance runs on a Dedicated Host, which is an isolated server with configurations that you can control.
            - Physical server with full EC2 capacity dedicated to the user. It's mostly used for Licensing reasons.

• Instance Life-cycle

- Pending
- Running **(Charges are incurred for every hour or partial hour the instance is running even if it is idle)**
- Start & Stop (EBS-backed instances only)
  - Charges are only incurred for the EBS storage and not for the instance hourly charge or data transfer but Charges for full hour are incurred for every transition from stopped to running.
  - While the instance is stopped, you can treat its root volume like any other volume, and modify it for e.g. repair file system problems or update software or change the instance type, user data, EBS otpmization attributes etc
  - An instance when stopped and started is launched on a new host
  - Any data on an instance store volume (not root volume) would be lost while data on the EBS volume persists
  - EC2 instance retains its private IP address as well as the Elastic IP address. However, the public IP address, if assigned instead of the Elastic IP address, would be released
- Instance reboot (Amazon recommends to use Amazon EC2 to reboot the instance instead of from the OS)
- Instance retirement (An instance is scheduled to be retired when AWS detects irreparable failure of the underlying hardware hosting the instance.)
- Instance Termination
  - Termination protection (DisableApiTermination attribute) can be enabled on the instance to prevent it from being accidentally terminated
  - Termination protection does not work for instances when
    - Part of an Autoscaling group
    - Launched as Spot instances
    - Terminating an instance by initiating shutdown from the instance
  - Data persistence
  - EBS volume have a DeleteOnTermination attribute which determines whether the volumes would be persisted or deleted when an instance they are associated with are terminated
  - Data on Instance store volume data does not persist
  - Data on EBS root volumes, have the DeleteOnTermination flag set to true, would be deleted by default
  - Additional EBS volumes attached have the DeleteOnTermination flag set to false are not deleted but just dettached from the instance

• EC2 VM Import/Export

EC2 VM Import/Export enables importing virtual machine (VM) images from existing virtualization environment to EC2, and then export them back

- Ability to import a VM from a virtualization environment to EC2 as an Amazon Machine Image (AMI) and EC2 and to import disks as Amazon EBS snapshots.

- ability to export a VM that was previously imported from the virtualization environment

• EC2 Key pairs

- EC2 uses public-key cryptography to encrypt & decrypt login information
- Public-key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data.
- Public and private keys are known as a **key pair.**
- To log in to an EC2 instance, a key pair needs to be created and specified when the instance is launched, and the private key can be used to connect to the instance. If you launch your instance without a key pair, you will not be able to access to it (via RDP or SSH)
- You can download the private key only once
- For Windows instances, the key pair can be used to obtain the administrator password and then log in using RDP
- EC2 stores the public key only, and the private key resides with the user. EC2 doesn't keep a copy of your private key
- EC2 Security Best Practice: Store the private keys in a secure place as anyone who possesses the private key can decrypt the login information
- **Also, if the private key is lost, there is no way to recover the same.**
  - For instance store, you cannot access the instance
  - For EBS-backed Linux instances, access can be regained.
    - EBS-backed instance can be stopped, its root volume detached and attached to another instance as a data volume
    - Modify the authorized_keys file, move the volume back to the original instance, and restart the instance
- **Key pair associated with the instances can either be**
  - **Generated by Amazon EC2**
    - Keys that Amazon EC2 uses are 2048-bit SSH-2 RSA keys.
  - **Created separately (using third-party tools) and Imported into EC2**
    - EC2 only accepts RSA keys and does not accept DSA keys
    - Supported lengths: 1024, 2048, and 4096

• Launching EC2 instances in a VPC

When launching an EC2 instance in a VPC you can:

- Leave it to AWS to select the best AZ for your EC2 instance (the recommended option) Or, specify which AZ you want your EC2 instance launched in
- Also, you can leave it to AWS to assign an IP address for your instance Or, you can assign the IP address (if you chose the subnet for the EC2 instance) in the subnet where you want to launch your EC2 instance
- You can create a public/private key pair
- Supports the following storage options: Amazon Elastic Block Sotre (EBS) , Amazon EC3 Instance Store and Amazon Simple Storage Service (S3)
- Linux, /dev/sda1 reserved for the root device

- Support s 2 types of block devices
- To launch an ec2 instance it is require to have an AMI in that region, if the AMI is not available in that region, then create a new AMI or use the copy command to copy the AMI from one region to the other region

• Placement Group

Let's you place multiple AWS EC2 instances in a group, this will provide a lower network latency.
AWS now provides two types of placement groups

- **Cluster** – clusters a logical grouping of instances into a low-network latency, high network throughput, or both in a single AZ
  - To provide the lowest latency, and the highest packet-per-second network performance for the placement group, choose an instance type that supports enhanced networking
  - recommended to launch all group instances at the same time to ensure enough capacity
  - instances can be added later, but there are chances of encountering an insufficient capacity error
  - For moving an instance into the placement group,
    - create an AMI from the existing instance,
    - and then launch a new instance from the AMI into a placement group.
  - Stopping and starting an instance within the placement group, the instance still runs in the same placement group
  - In case of an capacity error, stop and start all of the instances in the placement group, and try the launch again. Restarting the instances may migrate them to hardware that has capacity for all requested instances
  - It is more of an hint to AWS that the instances need to be launched physically close to each together
  - Enables applications to participate in a low-latency, 10 Gbps network.
- **Spread** – It is a group of instances that are each placed on distinct underlying hardware
  - Recommended for applications that have a small number of critical instances that should be kept separate from each other.
  - Reduces the risk of simultaneous failures that might occur when instances share the same underlying hardware.
  - Provide access to distinct hardware, and are therefore suitable for mixing instance types or launching instances over time.
  - Can span multiple AZs, and can have a maximum of seven running instances per AZ per group.
  - If the start or launch an instance in a spread placement group fails cause of insufficient unique hardware to fulfill the request, the request can be tried later as EC2 makes more distinct hardware available over time

Placement Group Rules and Limitations

- Ensure unique Placement group name within AWS account for the region

- Placement groups cannot be merged
- An instance can be launched in one placement group at a time; it cannot span multiple placement groups.
- Instances with a tenancy of host cannot be launched in placement groups.
    - **Cluster Placement group**
        - Only supported by Specific Instance types
        - maximum network throughput speed of traffic between two instances in a cluster placement group is limited by the slower of the two instances, so choose the instance type properly. (Up to 10 Gbps)
        - Traffic to and from S3 buckets within the same region over the public IP address space or through a VPC endpoint can use all available instance aggregate bandwidth.
        - recommended to use the same instance type
        - Network traffic to the internet and over an AWS Direct Connect connection to on-premises resources is limited to 5 Gbps.
    - **Spread placement groups**
        - They are not supported for Dedicated Instances or Dedicated Hosts.

• Bastion HOST

Centralize the remote access to a single EC2 instance in the company's AWS VPC then using that instance to connect to the other EC2 instances = bastion HOST

- Bastion means a structure for Fortification to protect things behind it
- In AWS, a Bastion host (also referred to as a Jump server) can be used to securely access instances in the private subnets.
- Bastion host launched in the Public subnets would act as a primary access point from the Internet and acts as a proxy to other instances.
-

Key points

- Bastion host is deployed in the Public subnet and acts as a proxy or a gateway between you and your instances
- Bastion host is a security measure that helps to reduce attack on your infrastructure and you have to concentrate to hardening a single layer
- Bastion host allows you to login to instances in the Private subnet securely without having to store the private keys on the Bastion host (using ssh-agent forwarding or RDP gateways)
- Bastion host security can be further tightened to allow SSH/RDP access from specific trusted IPs or corporate IP ranges
- Bastion host for your AWS infrastructure shouldn’t be used for any other purpose, as that could open unnecessary security holes
- Security for all the Instances in the private subnet should be hardened to accept SSH/RDP connections only from the Bastion host
- Deploy a Bastion host within each Availability Zone for HA, cause if the Bastion instance or the AZ hosting the Bastion server goes down the ability to connect to your private instances is lost completely

• EC2 with IAM Role

- IAM ROLE is something to attach to the EC2 instance to access other AWS services not to access the EC2 instance
- IAM policy can be defined to allow or deny a user access to the EC2 resources and actions
- IAM allows to control only what actions a user can perform on the EC2 resources but cannot be used to grant access for users to be able to access or login to the instances
- EC2 instances can be launched with IAM roles so that the applications can securely make API requests from your instances,
- IAM roles prevents the need to share as well as manage, rotate the security credentials that the applications use
- IAM role can be associated with the EC2 instance launched and to an existing running EC2 instance.
- To launch an instance with an IAM role, the name of its instance profile needs to be specified.
- Security credentials are temporary and are rotated automatically and new credentials are made available at least five minutes prior to the expiration of the old credentials.

Best Practice: Always launch EC2 instance with IAM role instead of hardcoded credentials

• AWS EC2 Best Practices

Security & Network

- Implement the least permissive rules for your security group.
- Regularly patch, update, and secure the operating system and applications on your instance
- Manage access to AWS resources and APIs using identity federation, IAM users, and IAM roles
- Establish credential management policies and procedures for creating, distributing, rotating, and revoking AWS access credentials

Storage

- EC2 supports Instance store and EBS volumes, so its best to understand the implications of the root device type for data persistence, backup, and recovery
- Use separate Amazon EBS volumes for the operating system (root device) versus your data.
- Ensure that the data volume (with your data) persists after instance termination
- Use the instance store available for your instance to only store temporary data. (Remember that the data stored in instance store is deleted when you stop or terminate your instance)
- If you use instance store for database storage, ensure that you have a cluster with a replication factor that ensures fault tolerance.

Resource Management

- Use instance metadata and custom resource tags to track and identify your AWS resources
- View your current limits for Amazon EC2. Plan to request any limit increases in advance of the time that you'll need them.

Backup & Recovery

- Regularly back up your instance using Amazon EBS snapshots (not done automatically) or a backup tool.
- Implement High Availability by deploying critical components of the application across multiple Availability Zones, and replicate the data appropriately
- Monitor and respond to events.
- Design your applications to handle dynamic IP addressing when your instance restarts.
- Implement failover. For a basic solution, you can manually attach a network interface or Elastic IP address to a replacement instance
- Regularly test the process of recovering your instances and Amazon EBS volumes if they fail.

• EC2 Monitoring - Impaired instance

(!) EC2 - Impaired instance, there is a software or hardware problem where the instance sits

Status Checks

- EC2 performs automated checks on every running EC2 instance to identify hardware and software issues.
- Status checks are performed every minute and each returns a pass or a fail status.
  - If all checks pass, the overall status of the instance is OK.
  - If one or more checks fail, the overall status is Impaired.
- Status checks are built into EC2, so they cannot be disabled or deleted.
- Alarms can be created, or deleted, that are triggered based on the result of the status checks. for e.g., an alarm can be created to warn if status checks fail on a specific instance.

System Status Checks

- Monitor the AWS systems required to use your instance to ensure they are working properly.
- When a system status check fails, one can either
  - Wait for AWS to fix the issue
  - Or resolve it by by stopping and restarting or terminating and replacing an instance

Instance Status Checks

- Monitor the software and network configuration of the individual instance
- When an instance status check fails, it can be resolved by either rebooting the instance or by making modifications in the operating system
-

CloudWatch Monitoring

- CloudWatch, helps monitor EC2 instances, which collects and processes
- raw data from EC2 into readable, near real-time metrics.

- By default Basic monitoring is enabled and EC2 metric data is sent to CloudWatch in 5-minute periods automatically
- Detailed monitoring can be enabled on EC2 instance, which sends data to CloudWatch in 1-minute periods.
- Aggregating Statistics Across Instances/ASG/AMI ID are available for the instances that have detailed monitoring (at an additional charge)

EC2 Metrics

- CPUCreditUsage
- CPU Credit metrics are available at a 5 minute frequency.
- CPUCreditBalance
- CPUUtilization
- DiskReadOps
- DiskWriteOps
- DiskReadBytes
- DiskWriteBytes
- NetworkIn
- NetworkOut
- NetworkPacketsIn
- NetworkPacketsOut
- StatusCheckFailed
- StatusCheckFailed_System
- StatusCheckFailed_Instance

• EC2 – Troubleshooting Instances

EC2 Troubleshooting An Instance Immediately Terminates

- EBS volume limit was reached. Its a soft limit and can be increased by submitting a support request
- EBS snapshot is corrupt.
- Instance store-backed AMI used to launch the instance is missing a required part

EC2 Troubleshooting Connecting to Your Instance

- **Network error: Connection timed out OR Error connecting to [instance] reason Connection timed out: connect**
  - **Reason:** Your connection request might not be reaching the Instance, or response might not be making it back to you (basically, connectivity is broken either by networking, security or CPU load)
  - **What to check/fix:**
    - Route table, for the subnet, does not have a route that sends all traffic destined outside the VPC to the Internet gateway for the VPC.
    - Security group does not allow inbound traffic from the public ip address on the proper port
    - ACL does not allow inbound traffic from and outbound traffic to the public ip address on the proper port
    - If connecting from an Corporate network, the internal firewall does not allow inbound and outbound traffic on port 22 (for Linux instances) or port 3389 (for Windows instances).
    - Instance does not have the same public IP address, which change during restarts. Associate an Elastic IP address with the instance

        - **High CPU load on the instance; the server may be overloaded.**
- **User key not recognized by server**
    - Private key file used to connect has not been converted to the format as required by the server
- **Host key not found in [directory], Permission denied (publickey) or Authentication failed, permission denied**
    - **Reason**
        - Your authentication have failed due to either:
            - Wron user-name AMI
            - Wrong private key
    - **What to check fix**
        - Verify your are using the correct user name for the AMI, for e.g. user name for Amazon Linux AMI is ec2-user, Ubuntu AMI is ubuntu, RHEL5 AMI & SUSE Linux can be either root or ec2-user, Fedora AMI can be fedora or ec2-user
        - Verify that you are using the correct private key file (.pem) which you created or used when you created the instance
- **Unprotected Private Key File**
    - **Reason:**
        - Private key file is not protected from read and write operations from any other users.
    - **What to check/fix:**
        - Use the chmod linux command to change the permissions on the file
- **Server refused our key or No supported authentication methods available**
    - **Reason:**
        - Appropriate user name for the AMI is not used for connecting

EC2 Troubleshooting Instances with Failed Status Checks

- System Status Check – Checks Physical Hosts
    - Lost of Network connectivity
    - Loss of System power
    - Software issues on physical host
    - Hardware issues on physical host
- **Resolution**
    - Amazon EBS-backed AMI instance – stop and restart the instance
    - Instance-store backed AMI – terminate the instance and launch a replacement.
- Instance Status Check – Checks Instance or VM
    - Possible reasons
        - Misconfigured networking or startup configuration
        - Exhausted memory
        - Corrupted file system
        - Failed Amazon EBS volume or Physical drive
        - Incompatible kernel
    - **Resolution**
        - Rebooting of the Instance or making modifications in your Operating system, volumes

EC2 Troubleshooting Instance Capacity

- **InsufficientInstanceCapacity**
  - AWS does not currently have enough available capacity to service your request. There is a limit for number of instances of instance type that can be launched within a region.
  - **To solve the issue, you can try the following steps:**
    - Wait a few minutes and then submit your request again
    - Submit a new request with a reduced number of instances
    - (if launching an Instance) Submit a new request without specifying an AZ or Submit a new request using a different instance type (which you can resize at a later stage)
    - Try purchasing Reserved Instances
- **InstanceLimitExceeded**
  - Concurrent running instance limit, default is 20, has been reached.

• EC2 Key points to remember

1. **Understand the difference between the three payment plans:**
   - **On-Demand instances:**
     - pay per hour
     - Use cases: development/ testing environment.
   - **Reserved instances**
     - 1 or 3-year agreement (up to 60% discount)
     - Use cases: production-ready application, applications that will be live for a long time.
   - **Spot Instances**
     - bid on the price
     - Use cases: applications that termination of an instance won't cause an issue.
2. Spot Instance: If the price of the spot instance goes above the bid price, or there is not enough capacity, the AWS EC2 instance will receive a termination notice and will be terminated in two minutes.
3. **There are three tenancy options:**
   - **Shared Tenancy**: Multiple instances run on the same hardware (default option)
   - **Dedicated instance**: A dedicated hardware that runs only a single customer instance
   - **Dedicated host**: Physical server with full EC2 capacity dedicated to the user. It's mostly used for Licensing reasons.
4. Instance store data is lost when an Amazon EC2 instance is restarted or terminated. It is temporary data!
5. **There are four types of AMI's**
   - AMI's that are published by AWS: These are maintained and managed by AWS and are very reliable.
   - AWS Marketplace: You could purchase AMI's from other providers such as Bitnami, Drupal. You don't need to install the applications.
   - Existing Instance AMI's: AMI's from a current EC2.
   - Uploaded AMI's from Virtual Servers: These are AMI's that have been Imported or Exported via AWS Import/Export Service.

6. volume can be mounted on multiple servers.
7. EBS volume can be mounted on a single server.
8. Public IP is changed on a stop/start of an instance. To avoid the change of IP, associate an Elastic IP to your instance.
9. Elastic IP attached to an EC2 instance will not incur any charge! But, if it is not associated you will be charged.
10. A newly launched Window Instance can be accessed via the Random Password which AWS generates upon the completion of the instance creation.
11. "**User data" field when launching an Instance.** Allows you to execute a script when an instance is booted. Usually, it involves installing certain packages or configuring Chef/Puppet. If you launch more than one instance at a time, the "user data" is available to all the instances in that reservation.
12. **Only 2 Access keys per user**
13. Partial instance-hours are billed as full hours
14. There is a 20 EC2 instance soft limit per account, you can submit a request to AWS to increase it
15. You should wait at least 20 minutes before open a ticket to AWS because there is problems that are resolved by thenself during that time
16. **Enhanced Networkin**g: Is a feature in AWS EC2 which improves the network connectivity. Note: Only specific EC2 types support it and can be enabled in a VPC only.
    - Enhanced networking results in higher bandwidth, higher packet per second (PPS) performance, lower latency, consistency, scalability and lower jitter
    - EC2 provides enhanced networking capabilities using single root I/O virtualization (SR-IOV) only on supported instance types
    - SR-IVO is a method of device virtualization which provides higher I/O performance and lower CPU utilization
    - Requirements:
        - HVM, Appropriate Virtual Function (VF) driver, Linux (ixgbevf module amd that **sriovNetSupport** attribute set for the instance
        - Supported instance types i.e. C3, C4, D2, I2, M4 and R3
        -

Amazon Storage services

- **Amazon EC2 provides flexible, cost effective and easy-to-use EC2 storage options with a unique combination of performance and durability**
    - Amazon Elastic Block Store (EBS)
    - Amazon EC2 Instance Store
    - Amazon Simple Storage Service (S3)
- While EBS and Instance Store are Block level, Amazon S3 is an Object level storage

Block Device Mapping

- A block device is a storage device that moves data in sequences of bytes or bits (blocks) and supports random access and generally use buffered I/O for e.g. hard disks, CD-ROM etc
- Block devices can be physically attached to a computer (like an instance store volume) or can be accessed remotely as if it was attached (like an EBS volume)
- Block device mapping defines the block devices to be attached to an instance, which can either be done while creation of an AMI or when an instance is launched
- Block device must be mounted on the instance, after being attached to the instance, to be able to be accessed
- When a block device is detached from an instance, it is unmounted by the operating system and you can no longer access the storage device.
- Additional Instance store volumes can be attached only when the instance is launched while EBS volumes can be attached to an running instance.
- View the block device mapping for an instance, only the EBS volumes can be seen, not the instance store volumes. Instance meta-data can be used to query the complete block device mapping.

• Elastic Block Storage – EBS

- Amazon EBS provides HA and reliability
- EBS as a primary storage device is recommended for data that requires frequent and granular updates for e.g. running a database or file-systems
- An EBS volume behaves like a raw, unformatted, external block device that can be attached to a single EC2 instance at a time
- EBS volume persists independently from the running life of an instance.
- An EBS volume can be attached to any instance within the same Availability Zone, and can be used like any other physical hard drive.
- EBS volumes allows encryption using the Amazon EBS encryption feature. All data stored at rest, disk I/O, and snapshots created from the volume are encrypted. Encryption occurs on the EC2 instance, providing encryption of data-in-transit from EC2 to the EBS volume
- EBS volumes can be backed up by creating a snapshot of the volume, which is stored in Amazon S3. EBS volumes can be created from a snapshot can be attached to an another instance within the same region
- EBS volumes are created in a specific Availability Zone, and can then be attached to any instances in that same Availability Zone. To make a volume available outside of the Availability Zone, create a snapshot and restore that snapshot to a new volume anywhere in that region
- Snapshots can also be copied to other regions and then restored to new volumes, making it easier to leverage multiple AWS regions for geographical expansion, data center migration, and disaster recovery.

Benefits

- **Data Availability**
  - EBS volume is automatically replicated in an Availability Zone to prevent data loss due to failure of any single hardware component.
- **Data Persistence**

    - EBS volume persists independently of the running life of an EC2 instance, it persists when an instance is stopped and started or rebooted
    - Root EBS volume is deleted, by default, on Instance termination but can be modified by changing the **DeleteOnTermination** flag
    - All attached volumes persist, by default, on instance termination
- **Data Encryption**
    - EBS volumes can be encrypted by EBS encryption feature
    - EBS encryption uses 256-bit Advanced Encryption Standard algorithms (AES-256) and an Amazon-Managed Key Infrastructure (KMS)
    - Encryption occurs on the server that hosts the EC2 instance, providing encryption of data-in-transit from the EC2 instance to EBS storage
    - Snapshots of encrypted EBS volumes are automatically encrypted.
- **Snapshots**
    - EBS provides the ability to create snapshots (backups) of any EBS volume and write a copy of the data in the volume to Amazon S3, where it is stored redundantly in multiple Availability Zones
    - Snapshots can be used to create new volumes, increase the size of the volumes or replicate data across Availability Zones
    - Snapshots are incremental backups and store only the data that was changed from the time the last snapshot was taken.
    - Snapshots size can probably be smaller then the volume size as the data is compressed before being saved to S3
    - Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.

EC2 block store devices

- **Two types of block store devices are supported:**
  - **Elastic Block Store (EBS)**
    - Persistent
    - Network attached virtual drives
    - You can resize your EBS volume size up but not down
  - **Instance-Store (Ephemeral store)**
    - Means the root volume is on the Instance-store disk image (ephemeral)
    - Limited to 10GB per device
    - **It does not have the STOP option**

Difference between Root/boot and backed volume

- **root volume:**
  - It contains the Root/boot volume (the disk where is installed the OS)/boot volumes can be EBS or Instance Store Volumes
    - It means it has an EBS root volume as the OS system (persistent)
- **non-root volume (backed)**:
  - Data Storage volume

• EBS Volume Types

The volumes types fall into two categories:

- **SSD-backed volumes** optimized for transactional workloads involving frequent read/write operations with small I/O size, where the dominant performance attribute is IOPS
- **HDD-backed** volumes optimized for large streaming workloads where throughput (measured in MiB/s) is a better performance measure than IOPS

	Solid-State Drives (SSD)	
Volume type	General Purpose SSD (gp2*)	Provisioned IOPS SSD (io1)
Description	General Purpose SSD volume that balances price and performance for a wide variety of workloads	Highest-Performance SSD volume for mission-critical low-latency or high throughput workloads
Use Cases	• Recommended for most workloads • System boot volumes • Virtual Desktops • Low-Latency interactive apps • Development and test environments	• Critical business applications that require sustained IOPS performance or more than 10000 IOPS or 160 MiB/s or throughput per volume • Large database workloads such as: ◦ MongoDB ◦ Casandra ◦ Microsoft SQL server ◦ MySQL ◦ PostgressSQL ◦ Oracle

API Name	gp2	io1
Volume Size	1 GIB – 16 TIB	4 GIB – 16 TIB
Max IOPS/Volume**	10.000	20,000/32,000
Max. Throughput/Volume	160 MIB/S	500 MIB/s
Max. IOPS/instance	80000	80000
Max. Throughput/Instance	1.750 MiB/s	1.750 MiB/s
Dominant Performance attribute	IOPS	IOPS

	Hard Disk Drives (HDD)	
Volume type	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Description	Low Cost HDD Volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
• **Use Cases**	• Streaming workloads requiring consistent, fast throughput at a low price • Big data • Data warehouses • Log processing • **Cannot be used as boot volume**	• Throughput-oriented storage for large volumes of data that is infrequently accessed • Scenarios where the lowest storage cost is important • **Cannot be used as boot volume**
API Name	st1	sc1
Volume Size	500 GIB – 16 TIB	500 GIB – 16 TIB
Max IOPS/Volume**	500	250
Max. Throughput/Volume	500 MIB/S	250 MIB/S
Max. IOPS/instance	80000	80000
Max. Throughput/Instance	1.750 MiB/s	1.750 MiB/s
Dominant Performance attribute	MIB/s	MIB/s

- **EBS Volume types keys**

- There are four types of EBS Volumes:
  - **Hard Disk Drives (HDD) (optimized for large streaming workloads where throughput (measured in MiB/s) is a better performance measure than IOPS)**
    - Cold HDD
      - Use case: applications with few data access.
      - Max IOPS 250.
    - Throughput Optimized HDD
      - Use case: data warehouse, log processing.
      - Max IOPS 500.
    - Magnetic (standard) HDD
      (Magnetic is a Previous Generation Volume. For new applications, we recommend using one of the newer volume types)
      - Use case: suited for workloads where data is accessed infrequently, and scenarios where low-cost storage for small volume sizes is important
      - Max IOPS 100
      - Volume size 1 GiB to 1 TiB
  - **Solid-State Drives (SSD) (optimized for transactional workloads IOPS)**
    - General Purpose SSD
      - Use case: OS boot volume, databases.
      - Max IOPS 10,000
    - Provisioned IOPS SSD.
      - Use case: Applications which need very fast data access, large databases
      - Max IOPS 20,000/32,000.

EBS volume can be created either

- Creating New volumes
  - Completely new from console or command line tools and can then be attached to an EC2 instance in the same Availability Zone
- Restore volume from Snapshots
  - EBS volumes can also be restored from a previously created snapshots
  - New volumes created from existing EBS snapshots load lazily in the background.
  - There is no need to wait for all of the data to transfer from S3 to the EBS volume before the attached instance can start accessing the volume and all its data.
  - If the instance accesses the data that hasn’t yet been loaded, the volume immediately downloads the requested data from Amazon S3, and continues loading the rest of the data in the background.
  - EBS volumes restored from encrypted snapshots are encrypted by default
- EBS volumes can be created and attached to a running EC2 instance by specifying a block device mapping

EBS Volume Detachment

- EBS volumes can be detached from an instance explicitly or by terminating the instance
- EBS root volumes can be detached by stopping the instance
- EBS data volumes, attached to an running instance, can be detached by unmounting the volume from the instance first. If the volume is detached without being unmounted, it might result the volume being stuck in the busy state and could possibly damaged the file system or the data it contains
- EBS volume can be force detached from an instance, using the **Force Detach option**, but it might lead to data loss or a corrupted file system as the instance does not get an opportunity to flush file system caches or file system metadata
- Charges are still incurred for the volume after its detachment

EBS Snapshots

- EBS provides the ability to create snapshots (backups) of any EBS volume and write a copy of the data in the volume to Amazon S3, where it is stored redundantly in multiple Availability Zones
- Snapshots can be used to create new volumes, increase the size of the volumes or replicate data across Availability Zones
- Snapshots are incremental backups and store only the data that was changed from the time the last snapshot was taken.
- Snapshots size can probably be smaller then the volume size as the data is compressed before being saved to S3
- Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.

EBS Snapshot creation

- Snapshots are incremental and only store the blocks on the device that changed since the last snapshot was taken
- Snapshots occur asynchronously; the point-in-time snapshot is created immediately while it takes time to upload the modified blocks to S3
- Snapshots can be taken from in-use volumes. However, snapshots will only capture the data that was written to the EBS volumes at the time snapshot command is issued excluding the data which is cached by any applications of OS
- Recommended ways to create a Snapshot from an EBS volume are:
  - Pause all file writes to the volume
  - Unmount the Volume -> Take Snapshot -> Remount the Volume
  - Stop the instance – Take Snapshot (for root EBS volumes)
  - Snapshots of encrypted volumes are encrypted and volumes created from encrypted snapshots are automatically encrypted

EBS Snapshot Deletion

- When a snapshot is deleted only the data exclusive to that snapshot is removed.

- Deleting previous snapshots of a volume do not affect your ability to restore volumes from later snapshots of that volume.
- Active snapshots contain all of the information needed to restore your data (from the time the snapshot was taken) to a new EBS volume.
- Even though snapshots are saved incrementally, the snapshot deletion process is designed so that you need to retain only the most recent snapshot in order to restore the volume.
- Snapshot of the root device of an EBS volume used by a registered AMI can't be deleted. AMI needs to be deregistered to be able to delete the snapshot.

EBS Snapshot Copy

- Snapshots are constrained to the region in which they are created and can be used to launch EBS volumes within the same region only
- Snapshots can be copied across regions to make it easier to leverage multiple regions for geographical expansion, data center migration, and disaster recovery
- Snapshots are copied with S3 server-side encryption (256-bit Advanced Encryption Standard) to encrypt your data and the snapshot copy receives a snapshot ID that's different from the original snapshot's ID.
- User-defined tags are not copied from the source to the new snapshot.
- First Snapshot copy to another region is always a full copy, while the rest are always incremental.
- When a snapshot is copied, it can be encrypted if currently unencrypted or can be encrypted using a different encryption key. Changing the encryption status of a snapshot or using a non-default EBS CMK during a copy operation always results in a full copy (not incremental)

EBS Snapshot Sharing

- Snapshots can be shared by making them public or with specific AWS accounts by modifying the permissions of the snapshots
- Encrypted snapshot can be shared with specific AWS accounts, though you cannot make it public. For others to use the snapshot, you must also share the custom CMK key used to encrypt it. Cross-account permissions may be applied to a custom key either when it is created or at a later time. Users with access can copy your snapshot and create their own EBS volumes based on your snapshot while your original snapshot remains unaffected.
- AWS prevents you from sharing snapshots that were encrypted with your default CMK

• EBS Encryption

- EBS volumes can be created and attached to a supported instance type, and supports following types of data
  - Data at rest
  - All snapshots created from the volume
  - All disk I/O
- Encryption occurs on the servers that host EC2 instances, providing encryption of data-in-transit from EC2 instances to EBS storage.

- EBS encryption is supported with all EBS volume types (gp2, io1 and standard), and has the same IOPS performance on encrypted volumes as with unencrypted volumes, with a minimal effect on latency
- EBS encryption is only available on select instance types
- Snapshots of encrypted volumes and volumes created from encrypted snapshots are automatically encrypted using the same volume encryption key
- EBS encryption uses AWS Key Management Service (AWS KMS) customer master keys (CMK) when creating encrypted volumes and any snapshots created from the encrypted volumes.
- **EBS volumes can be encrypted using either:**
  - A default CMK is created for you automatically.
  - A CMK that you created separately using AWS KMS (under IAM section), giving you more flexibility, including the ability to create, rotate, disable, define access controls, and audit the encryption keys used to protect your data.
- Public or shared snapshots of encrypted volumes are not supported, because other accounts would be able to decrypt your data and needs to be migrated to an unencrypted status before sharing.
- Existing unencrypted volumes cannot be encrypted directly, but can be migrated by
  - Create a unencrypted snapshot from the volume
  - Create an encrypted copy of unencrypted snapshot
  - Create an encrypted volume from the encrypted snapshot
- Encrypted snapshot can be created from a unencrypted snapshot by create an encrypted copy of the unencrypted snapshot
- Unencrypted volume cannot be created from an encrypted volume directly but needs to be migrated

• EBS Performance Tips

- EBS Performance depends on several factors including I/O characteristics and the configuration of instances and volumes and can be improved using Provisioned IOPS, EBS-Optimized instances, Pre-Warming and RAIDed configuration

• EBS-Optimized or 10 Gigabit Network Instances

- An EBS-Optimized instance uses an optimized configuration stack and provides additional, dedicated capacity for EBS I/O.
- Optimization provides the best performance for the EBS volumes by minimizing contention between EBS I/O and other traffic from a instance.
- EBS-Optimized instances deliver dedicated throughput to EBS, with options between 500 Mbps and 4,000 Mbps, depending on the instance type used
- Not all instance types support EBS-Optimization
- Some Instance type enable EBS-Optimization by default, while it can be enabled for some.
- EBS optimization enabled for an instance, that is not EBS-Optimized by default, an additional low, hourly fee for the dedicated capacity is charged
- When attached to an EBS–optimized instance,

- General Purpose (SSD) volumes are designed to deliver within 10% of their baseline and burst performance 99.9% of the time in a given year
- Provisioned IOPS (SSD) volumes are designed to deliver within 10% of their provisioned performance 99.9 percent of the time in a given year.

EBS Volume Initialization – Pre-warming

- New EBS volumes receive their maximum performance the moment that they are available and DO NOT require initialization (pre-warming).
- EBS volumes needed a pre-warming, previously, before being used to get maximum performance to start with. Pre-warming of the volume was possible by writing to the entire volume with 0 for new volumes or reading entire volume for volumes from snapshots
- Storage blocks on volumes that were restored from snapshots must be initialized (pulled down from S3 and written to the volume) before the block can be accessed
- This preliminary action takes time and can cause a significant increase in the latency of an I/O operation the first time each block is accessed.

• RAID Configuration

- EBS volumes can be striped, if a single EBS volume does not meet the performance and more is required.
- Striping volumes allows pushing tens of thousands of IOPS.
- EBS volumes are already replicated across multiple servers in an AZ for availability and durability, so AWS generally recommend striping for performance rather than durability.
- For greater I/O performance than can be achieved with a single volume, RAID 0 can stripe multiple volumes together; for on-instance redundancy, RAID 1 can mirror two volumes together.
- RAID 0 allows I/O distribution across all volumes in a stripe, allowing straight gains with each addition.
- RAID 1 can be used for durability to mirror volumes, but in this case, it requires more EC2 to EBS bandwidth as the data is written to multiple volumes simultaneously and should be used with EBS–optimization.
- EBS volume data is replicated across multiple servers in an AZ to prevent the loss of data from the failure of any single component
- AWS doesn’t recommend RAID 5 and 6 because the parity write operations of these modes consume the IOPS available for the volumes and can result in 20-30% fewer usable IOPS than a RAID 0.
- A 2-volume RAID 0 config can outperform a 4-volume RAID 6 that costs twice as much.

Configuration	Use	Advantages	Disadvantages
RAID 0	When I/O performance is more important than fault tolerance; for example, as in a heavily used database (where data replication is already set up separately).	I/O is distributed across the volumes in a stripe. If you add a volume, you get the straight addition of throughput.	Performance of the stripe is limited to the worst performing volume in the set. Loss of a single volume results in a complete data loss for the array.
RAID 1	When fault tolerance is more important than I/O performance; for example, as in a critical application.	Safer from the standpoint of data durability.	Does not provide a write performance improvement; requires more Amazon EC2 to Amazon EBS bandwidth than non-RAID configurations because the data is written to multiple volumes simultaneously.

• Other EBS Key Notes

- IOPs 10% from the Volume Size, of 1000 rest 100 and from 10 1, from 1TB it is the 99.9% IOPS (900GB)
- Amazon EBS measures each I/O operation per second → 256KB = 1 IOPS
- Recommended conduct snapshots to amazon s3
- Decommissioning process – Amazon destroy Hard Disks using the **DoD 52200.22-M or NIST 800-88** technique to destroy data as part of the decommissioning

• AWS EC2 Instance Store Storage

- An instance store provides temporary block-level storage for an EC2 instance.
- Instance store storage is located on the disks that are physically attached to the host computer.
- Instance store is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers.
- An instance store consists of one or more instance store volumes exposed as block devices.
- Virtual devices for instance store volumes are ephemeral[0-23], starting the first one as ephemeral0 and so on.
- While an instance store is dedicated to a particular instance, the disk subsystem is shared among instances on a host computer.

Instance Store Lifecycle

- Data on an instance store volume persists only during the lifetime of the associated instance; if an instance is stopped or terminated, any data on instance store volumes is lost.
- Instance store backed Instances cannot be stopped, as when stopped and started AWS does not guarantee the instance would be launched in the same host and hence the data is lost
- Data on Instance store volume is LOST in following scenarios :-

    - Failure of an underlying drive
    - Stopping an EBS-backed instance where instance store are attached as additional volumes
    - Termination of the Instance
- Data on Instance store volume is NOT LOST when the instance is rebooted
- If an AMI is created from an Instance with Instance store volume, the data on its instance store volume isn't preserved

Instance Store Volumes

- Instance type of an instance determines the size of the instance store available for the instance, and the type of hardware used for the instance store volumes
- Instance store volumes are included as part of the instance's hourly cost.
- Some instance types use solid state drives (SSD) to deliver very high random I/O performance, which is a good option when storage with very low latency is needed, but the data does not need to be persisted when the instance terminates or you can take advantage of fault tolerant architectures.

Instance Store Volumes with EC2 instances

- EBS volumes and instance store volumes for an instance are specified using a block device mapping
- Instance store volume can be attached to an EC2 instance only when the instance is launched
- Instance store volume cannot be detached and reattached to a different instance
- After an instance is launched, the instance store volumes for the instance should be formatted and mounted before it can be used.
- Root volume of an instance store-backed instance is mounted automatically.

• Instance Store (Ephemeral Storage Key-points)

- An Instance store backed instance is an EC2 instance using an Instance store as root device volume created from a template stored in S3.
- Boot time is slower then EBS backed volumes and usually less then 5 min
- Can be selected as Root Volume and attached as additional volumes
- Instance store backed Instances can be of maximum 10GiB volume size
- Instance store volume can be attached as additional volumes only when the instance is being launched and cannot be attached once the Instance is up and running
- For EC2 instance store-backed instances AWS recommends to:
    - Distribute the data on the instance stores across multiple Azs
    - Back up critical data from the instance store volumes to persistent storage on a regular basis.
    - Instance store backed Instances cannot be upgraded

Characteristic	Amazon EBS-Backed	Amazon Instance Store-Backed
Boot time	Usually less than 1 minute	Usually less than 5 minutes
Size limit	16 TiB	10 GiB
Root device volume	Amazon EBS volume	Instance store volume
Data persistence	By default, the root volume is deleted when the instance terminates.* Data on any other Amazon EBS volumes persists after instance termination by default. Data on any instance store volumes persists only during the life of the instance.	Data on any instance store volumes persists only during the life of the instance. Data on any Amazon EBS volumes persists after instance termination by default.
Upgrading	The instance type, kernel, RAM disk, and user data can be changed while the instance is stopped.	Instance attributes are fixed for the life of an instance.
Charges	You're charged for instance usage, Amazon EBS volume usage, and storing your AMI as an Amazon EBS snapshot.	You're charged for instance usage and storing your AMI in Amazon S3.
AMI creation/bundling	Uses a single command/call	Requires installation and use of AMI tools
Stopped state	Can be placed in stopped state where instance is not running, but the root volume is persisted in Amazon EBS	Cannot be in stopped state; instances are running or terminated

* By default, Amazon EBS-backed instance root volumes have the `DeleteOnTermination` flag set to `true`. For information about how to change this flag so that the volume persists after termination, see Changing the Root Device Volume to Persist.

- ## Amazon Elastic File System (Amazon EFS)

Amazon EFS provides scalable file storage for use with Amazon EC2. You can create an EFS file system and configure your instances to mount the file system. You can use an EFS file system as a common data source for workloads and applications running on multiple instances. For example, you can create an EFS file system on two Linux instances that can share data using the file system.

Note: Amazon EFS is not supported on Windows instances.

- ## An USER CASE: Where to store/upload images? S3 OR EFS?

When users upload images, you can't save the files locally. The images have to be stored in a location where all the EC2 instances have access.
Two AWS services that we can be used for this purpose:

- **Simple Storage Service (S3)**
- **Elastic File System (EFS).**

Simple Storage Service (S3

S3 is an object storage service from AWS where you can store a massive amount of data easily. You pay per GB of storage without any minimum. It's used by Netflix to store billions of hours of contents and by Airbnb to store 10 petabytes of user pictures. You never have to worry about capacity planning. They'll always have room to store your images.

- Most gems support S3 as their storage so you don't actually have to do a lot of work.
- If your Rails app store the images on S3, it doesn't matter which EC2 instance processes the image. They'll all store the images on the same S3 bucket.
- After making these changes, you'll use Carrierwave the same way as when storing the images on the local filesystem.
- How to do it step by step:
  - Go to the S3 Console and create a bucket. Enter a name and leave all the default permissions including "Do not grant public read access to this bucket (Recommended)".
  - Create a Policy on the IAM Console. Click Policies, Create Policy, and Create Your Own Policy. Enter a policy name and put the code below on the policy document. Change BUCKETNAME with the name of the bucket you created on Step 1.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "s3:*",
         "Resource": [
            "arn:aws:s3:::BUCKETNAME/*"
         ]
      }
   ]
}
```

  - Create an IAM user on the IAM Console. Click Users, Add User. Enter the name and check the box "Programmatic access. Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools."
  - On the Permissions step, click "Attach existing policies directly". Choose the policy you created on Step 2.

Elastic File System (EFS)

Elastic File System provides a shared filesystem that can be used from multiple EC2 instances at the same time. Remember when we said we can't use a local filesystem to store our images? That's still true but we can use a shared filesystem to get around this. We can mount the shared filesystem to /images for example. Then we'll symlink public/uploads to /images/uploads on all EC2 instances.

- Since /images is on EFS, all EC2 instances will see the same public/uploads.
- Follow the EFS Documentation to create a filesystem. On your EC2 instances, mount the filesystem using the standard Linux mount command.

Summary

- It is recommended to use S3 as it's more tested at this point. EFS is still a good option and may simplify your set-up so give it a try too.
- You can use S3 even if your Rails servers are not hosted on AWS. All you need is an S3 bucket and IAM keys. It looks like you may also be able to use EFS outside of AWS with some workarounds but the S3 setup is more common.

Amazon S3

- **Amazon S3 is a simple key, value object store designed for the Internet**
- S3 provides unlimited storage space and works on the pay as you use model. Service rates gets cheaper as the usage volume increases
- S3 is an Object level storage (not a Block level storage) and cannot be used to host OS or dynamic websites
- S3 resources for e.g. buckets and objects are private by default

S3 Buckets & Objects

- **Buckets**
  - A bucket is a container for objects stored in S3
  - S3 bucket names are globally unique, regardless of the AWS region in which you create the bucket
  - Even though S3 is a global service, buckets are created within a region specified during the creation of the bucket
  - There is no limit to the number of objects that can be stored in a bucket and no difference in performance whether you use many buckets to store your objects or a single bucket to store all your objects
  - S3 data model is a flat structure i.e. there are no hierarchies or folders within the buckets. However, logical hierarchy can be inferred using the keyname prefix e.g. Folder1/Object1
  - Restrictions
    - 100 buckets (soft limit) can be created in each of AWS account
    - Bucket names should be globally unique and DNS compliant
    - Bucket ownership is not transferable
    - Buckets cannot be nested and cannot have bucket within another bucket
  - You can delete an empty or a non-empty bucket
  - S3 allows retrieval of 1000 objects and provides pagination support

- **Objects**
  - Object is uniquely identified within a bucket by a keyname and version ID
  - Objects consist of object data, metadata and others
    - **Key** is object name
    - **Value** is data portion is opaque to S3
    - **Metadata** is the data about the data and is a set of name-value pairs that describe the object for e.g. content-type, size, last

modified. Custom metadata can also be specified at the time the object is stored.
        - **Version ID** is the version id for the object and in combination with the key helps to unique identify an object within a bucket
        - **Subresources** helps provide additional information for an object
        - **Access Control** Information helps control access to the objects
    - Metadata for an object cannot be modified after the object is uploaded and it can be only modified by performing the copy operation and setting the metadata
    - Objects belonging to a bucket reside in a specific AWS region never leave that region, unless explicitly copied using Cross Region replication
    - Object can be retrieved as a whole or a partially
    - With Versioning enabled, you can retrieve current as well as previous versions of an object

- **Bucket & Object Operations**
    - **Listing**
        - S3 allows listing of all the keys within a bucket
        - A single listing request would return a max of 1000 object keys with pagination support using an indicator in the response to indicate if the response was truncated
        - Keys within a bucket can be listed using Prefix and Delimiter.
        - Prefix limits results to only those keys (kind of filtering) that begin with the specified prefix, and delimiter causes list to roll up all keys that share a common prefix into a single summary list result.
    - **Retrieval**
        - Object can be retrieved as a whole
        - Object can be retrieved in parts or partially (specific range of bytes) by using the Range HTTP header.
        - Range HTTP header is helpful:
            - if only partial object is needed for e.g. multiple files were uploaded as a single archive
            - For fault tolerant downloads where the network connectivity is poor
        - Objects can also be downloaded by sharing Pre-Signed urls
        - Metadata of the object is returned in the response headers
    - **Object Uploads**
        - Single Operation – Objects of size 5GB can be uploaded in a single PUT operation
        - Multipart upload – can be used for objects of size > 5GB and supports max size of 5TB can is recommended for objects above size 100MB
        - Pre-Signed URLs can also be used shared for uploading objects
        - Uploading object if successful, can be verified if the request received a success response. Additionally, returned ETag can be compared to the calculated MD5 value of the upload object
    - **Copying Objects**
        - Copying of object up to 5GB can be performed using a single operation and multipart upload can be used for uploads up to 5TB
        - When an object is copied:

        - User-controlled system metadata e.g. storage class and user-defined metadata are also copied.
        - System controlled metadata e.g. the creation date etc is reset
    - Copying Objects can be needed
        - Create multiple object copies
        - Copy object across locations
        - Renaming of the objects
        - Change object metadata for e.g. storage class, server-side encryption etc
        - Updating any metadata for an object requires all the metadata fields to be specified again
- **Deleting Objects**
    - S3 allows deletion of a single object or multiple objects (max 1000) in a single call
    - For Non Versioned buckets,
        - The object key needs to be provided and object is permanently deleted
    - **For Versioned buckets,**
        - if an object key is provided, S3 inserts a delete marker and the previous current object becomes non current object
        - if an object key with a version ID is provided, the object is permanently deleted
        - if the version ID is of the delete marker, the delete marker is removed and the previous non current version becomes the current version object
    - Deletion can be MFA enabled for adding extra security
- **Restoring Objects from Glacier**
    - Objects must be restored before you can access an archived object
    - Restoration request also needs to specify the number of days for which the object copy needs to be maintained.
    - During this period, the storage cost for both the archive and the copy is charged

Pre-Signed URLs

- **All buckets and objects are by default private**
- Pre-signed URLs allows user to be able to download or upload a specific object without requiring AWS security credentials or permissions
- Pre-signed URL allows anyone access to the object identified in the URL, provided the creator of the URL has permissions to access that object
- Creation of the pre-signed urls requires the creator to provide his security credentials, specify a bucket name, an object key, an HTTP method (GET for download object & PUT of uploading objects), and expiration date and time
- Pre-signed urls are valid only till the expiration date & time

• Multipart Upload

- Multipart upload allows the user to upload a single object as a set of parts. Each part is a contiguous portion of the object's data.
- Multipart uploads supports 1 to 10000 parts and each Part can be from 5MB to 5GB with last part size allowed to be less than 5MB
- Multipart uploads allows max upload size of 5TB
- Object parts can be uploaded independently and in any order. If transmission of any part fails, it can be retransmitted without affecting other parts.
- After all parts of the object are uploaded and complete initiated, S3 assembles these parts and creates the object.
- Using multipart upload provides the following advantages:
  - Improved throughput – parallel upload of parts to improve throughput
  - **Quick recovery from any network issues – Smaller part size minimizes the impact of restarting a failed upload due to a network error.**
  - Pause and resume object uploads – Object parts can be uploaded over time. Once a multipart upload is initiated there is no expiry; you must explicitly complete or abort the multipart upload.
  - Begin an upload before the final object size is known – an object can be uploaded as is it being created
- Three Step process:
- **Multipart Upload Initiation**
  - Initiation of a Multipart upload request to S3 returns a unique ID for each multipart upload.
  - This ID needs to be provided for each part uploads, completion or abort request and listing of parts call.
  - All the Object metadata required needs to be provided during the Initiation call
- **Parts Upload**
  - Parts upload of objects can be performed using the unique upload ID
  - A part number (between 1 – 10000) needs to be specified with each request which identifies each part and its position in the object
  - If a part with the same part number is uploaded, the previous part would be overwritten
  - After the part upload is successful, S3 returns an ETag header in the response which must be recorded along with the part number to be provided during the multipart completion request
- **Multipart Upload Completion or Abort**
  - On Multipart Upload Completion request, S3 creates an object by concatenating the parts in ascending order based on the part number and associates the metadata with the object
  - Multipart Upload Completion request should include the unique upload ID with all the parts and the ETag information
  - S3 response includes an ETag that uniquely identifies the combined object data
  - On Multipart upload Abort request, the upload is aborted and all parts are removed. Any new part upload would fail. However, any in progress part upload is completed and hence and abort request must be sent after all the parts upload have been completed

        - S3 should receive a multipart upload completion or abort request else it will not delete the parts and storage would be charged

• Virtual Hosted Style vs Path-Style Request

S3 allows the buckets and objects to be referred in Path-style or Virtual hosted-style URLs

- **Path-style**
  Bucket name is not part of the domain (unless you use a region specific endpoint)
  - The endpoint used must match the region in which the bucket resides
  - for e.g, if you have a bucket called mybucket that resides in the EU (Ireland) region with object named puppy.jpg, the correct path-style syntax URI is
  - A **"PermanentRedirect"** error is received with an HTTP response code 301, and a message indicating what the correct URI is for the resource if a bucket is accessed outside the US East (N. Virginia) region with path-style syntax that uses either of the following:
    - http://s3.amazonaws.com
    - An endpoint for a region different from the one where the bucket resides. For example, if you use http://s3-eu-west-1.amazonaws.com for a bucket that was created in the US West (N. California) region
- **Virtual hosted-style**
  - S3 supports virtual hosted-style and path-style access in all regions.
  - In a virtual-hosted-style URL, the bucket name is part of the domain name in the URL
  - for e.g. http://bucketname.s3.amazonaws.com/objectname
  - S3 virtual hosting can be used to address a bucket in a REST API call by using the HTTP Host header
  - **Benefits**
    - Attractiveness of customized URLs,
    - Provides an ability to publish to the "root directory" of the bucket's virtual server. This ability can be important because many existing applications search for files in this standard location.
- S3 updates DNS to reroute the request to the correct location when a bucket is created in any region, which might take time.
- S3 routes any virtual hosted-style requests to the US East (N.Virginia) region, by default, if the US East (N. Virginia) endpoint s3.amazonaws.com is used, instead of the region-specific endpoint (for example, s3-eu-west-1.amazonaws.com) and S3 redirects it with HTTP 307 redirect to the correct region.
- When using virtual hosted-style buckets with SSL, the SSL wild card certificate only matches buckets that do not contain periods. To work around this, use HTTP or write your own certificate verification logic.
- If you make a request to the http://bucket.s3.amazonaws.com endpoint, the DNS has sufficient information to route your request directly to the region where your bucket resides.

S3 Pricing

- Amazon S3 costs vary by Region
- Charges in S3 are incurred for:
  - **Storage –** cost is per GB/month
  - **Requests** – per request cost varies depending on the request type GET, PUT
  - **Data Transfer**
    - Data transfer **IN** is free
    - Data transfer **OUT** is charged per GB/month (except in the same region or to Amazon CloudFront)

• S3 Storage Classes Overview

S3 storage classes allows life-cycle management for automatic migration of objects for cost savings

- There are three S3 storage classes:
  - **S3 Standard** (it is the default storage class):
    - Amazon S3 Standard offers high durability, availability, and performance object storage for frequently accessed data.
    - 99.99% availability and 99.999999999% durability
    - Supports SSL for data in transit and encryption of data at rest
    - Use at least two AZs
    - Use cases including cloud applications, dynamic websites, content distribution, mobile and gaming applications, and Big Data analytics
  - **S3 STANDARD_IA** (Infrequent Access) storage class:
    - STANDARD_IA objects are available for real-time access.
    - STANDARD_IA storage class is suitable for larger objects greater than 128 KB (smaller objects are charged for 128KB only) kept for at least 30 days.
    - Same low latency and high throughput performance of Standard
      - **S3 Standard-Infrequent Access**:
        - Amazon S3 Standard-Infrequent Access (S3 Standard-IA) is an Amazon S3 storage class for data that is accessed less frequently, but requires rapid access when needed. S3
        - Ideal for long-term storage, backups, and as a data store for disaster recovery and older data where access is limited, but the use case still demands high performance
        - Use at least two AZs
        - 99.9% availability and 99.999999999% durability
        - Supports SSL for data in transit and encryption of data at rest
      - **S3 Standard Infrequent Access ONEZONE** (less expensive):
        99.5% availability and 99.999999999% durability

Not resilient to the loss of the Availability Zone.

- **S3 RRS** (Reduced Redundancy Storage):
  - It is designed for non-critical, reproducible data stored at lower levels of redundancy than the STANDARD storage class, which reduces storage costs
  - Cheapest. ideal for objects that data loss is not important. (It is recommended that you not use this storage class. The STANDARD storage class is more cost effective.)
  - 99.99% availability and 99.99% durability
  - RRS stores objects on multiple devices across multiple facilities
  - If an RRS object is lost, S3 returns a 405 error on requests made to that object
  - S3 can send an event notification, configured on the bucket, to alert a user or start a workflow when it detects that an RRS object is lost which can be used to replace the lost object
  -

• Life-cycle Policy for an S3 Bucket

A life-cycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:
You can use it do delete data using the expiration time (for example when uploading logs to a bucket).
It can be configured using an XML file or through console interface (web):
Examples:

- Transition to Standard-IA after, and then type the number of days (for example, 30 days).
- Transition to One Zone-IA after, and then type the number of days (for example, 30 days).
- Transition to Amazon Glacier after, and then type the number of days (for example 100 days).

Amazon S3 stores the archived objects in Amazon Glacier. However, these are Amazon S3 objects, and you can access them only by using the Amazon S3 console or the Amazon S3 API.

• AWS S3 Data Protection (Encryption)

- Objects are redundantly stored on multiple devices across multiple facilities in an S3 region.
- Amazon S3 PUT and PUT Object copy operations synchronously store the data across multiple facilities before returning SUCCESS.
- Once the objects are stored, S3 maintains its durability by quickly detecting and repairing any lost redundancy.
- S3 also regularly verifies the integrity of data stored using checksums. If Amazon S3 detects data corruption, it is repaired using redundant data.

- In addition, S3 calculates checksums on all network traffic to detect corruption of data packets when storing or retrieving data
- Data protection against accidental overwrites and deletions can be added by enabling Versioning to preserve, retrieve and restore every version of the object stored
- S3 also provides the ability to protect data in-transit (as it travels to and from S3) and at rest (while it is stored in S3)

Data Protection

- **Data in-transit**
  - S3 allows protection of data in-transit by enabling communication via SSL or using client-side encryption

- **Data at Rest**
  - S3 supports both client side encryption and server side encryption for protecting data at rest
  - Using Server-Side Encryption, S3 encrypts the object before saving it on disks in its data centers and decrypt it when the objects are downloaded
  - Using Client-Side Encryption, you can encrypt data client-side and upload the encrypted data to S3. In this case, you manage the encryption process, the encryption keys, and related tools.

- **Server-Side Encryption**
  - Server-side encryption is about data encryption at rest
  - Server-side encryption encrypts only the object data. Any object metadata is not encrypted.
  - S3 handles the encryption (as it writes to disks) and decryption (when you access the objects) of the data objects
  - There is no difference in the access mechanism for both encrypted or unencrypted objects and is handled transparently by S3

  - **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)**
    - Each object is encrypted with a unique data key employing strong multi-factor encryption.
    - SSE-S3 encrypts the data key with a master key that is regularly rotated.
    - S3 server-side encryption uses one of the strongest block ciphers available , 256-bit Advanced Encryption Standard (AES-256), to encrypt the data.
    - Whether or not objects are encrypted with SSE-S3 can’t be enforced when they are uploaded using pre-signed URLs, because the only way you can specify server-side encryption is through the AWS Management Console or through an HTTP request header

- **Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)**
  - SSE-KMS is similar to SSE-S3, but it uses **AWS Key management Services (KMS)** which provides additional benefits along with additional charges
    - KMS is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud.
    - KMS uses customer master keys (CMKs) to encrypt the S3 objects.
    - Master key is never made available
    - KMS enables you to centrally create encryption keys, define the policies that control how keys can be used
    - Allows audit use of key usage to prove they are being used correctly, by inspecting logs in AWS CloudTrail
    - Allows keys to temporarily disabled and re-enabled
    - Allows keys to be rotated regularly
    - Security controls in AWS KMS can help meet encryption-related compliance requirements.
  - SSE-KMS enables separate permissions for the use of an envelope key (that is, a key that protects the data's encryption key) that provides added protection against unauthorized access of the objects in S3.
  - SSE-KMS provides the option to create and manage encryption keys yourself, or use a default customer master key (CMK) that is unique to you, the service you're using, and the region you're working in.
  - Creating and Managing your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.
  - Data keys used to encrypt your data are also encrypted and stored alongside the data they protect and are unique to each object
  - Process flow
    - An application or AWS service client requests an encryption key to encrypt data and passes a reference to a master key under the account.
    - Client requests are authenticated based on whether they have access to use the master key.
    - A new data encryption key is created, and a copy of it is encrypted under the master key.
    - Both the data key and encrypted data key are returned to the client.
    - Data key is used to encrypt customer data and then deleted as soon as is practical.
    - Encrypted data key is stored for later use and sent back to AWS KMS when the source data needs to be decrypted.

- **Server-Side Encryption usingCustomer-Provided Keys (SSE-C)**
  - Encryption keys can be managed and provided by the Customer and S3 manages the encryption, as it writes to disks, and decryption, when you access the objects

        - When you upload an object, the encryption key is provided as a part of the request and S3 uses that encryption key to apply AES-256 encryption to the data and removes the encryption key from memory.
        - When you download an object, the same encryption key should be provided as a part of the request. S3 first verifies the encryption key and if matches decrypts the object before returning back to you
        - As each object and each object's version can be encrypted with a different key, you are responsible for maintaining the mapping between the object and the encryption key used.
        - SSE-C request must be done through HTTPS and S3 will reject any requests made over http when using SSE-C.
        - For security considerations, AWS recommends to consider any key sent erroneously using http to be compromised and discarded or rotated
        - S3 does not store the encryption key provided. Instead, it stores a randomly salted HMAC value of the encryption key which can be used to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means, if you lose the encryption key, you lose the object.

Client-Side Encryption

Client-side encryption refers to encrypting data before sending it to Amazon S3 and decrypting the data after downloading it

- **AWS KMS-managed customer master key (CMK)**
    - Customer can maintain the encryption CMK with AWS KMS and can provide the CMK id to the client to encrypt the data
    - Uploading Object
        - AWS S3 encryption client first sends a request to AWS KMS for the key to encrypt the object data
        - AWS KMS returns a randomly generated data encryption key with 2 versions a plain text version for encrypting the data and cipher blob to be uploaded with the object as object metadata
        - Client obtains a unique data encryption key for each object it uploads.
        - AWS S3 encryption client uploads the encrypted data and the cipher blob with object metadata
    - Download Object
        - AWS Client first downloads the encrypted object from Amazon S3 along with the cipher blob version of the data encryption key stored as object metadata.
        - AWS Client then sends the cipher blob to AWS KMS to get the plain text version of the same, so that it can decrypt the object data.

- **Client-Side master key**

Encryption master keys are completely maintained at Client-side

    - Uploading Object

        - Amazon S3 encryption client (for e.g. AmazonS3EncryptionClient in the AWS SDK for Java) locally generates randomly a one-time-use symmetric key (also known as a data encryption key or data key).
        - Client encrypts the data encryption key using the customer provided master key
        - Client uses this data-encryption key to encrypt the data of a single S3 object (for each object, the client generates a separate data key).
        - Client then uploads the encrypted data to Amazon S3 and also saves the encrypted data key and itsmaterial description as object metadata (x-amz-meta-x-amz-key) in Amazon S3 by default
    - Downloading Object
        - Client first downloads the encrypted object from Amazon S3 along with the object metadata.
        - Using the material description in the metadata, the client first determines which master key to use to decrypt the encrypted data key.
        - Using that master key, the client decrypts the data key and uses it to decrypt the object
    - Client-side master keys and your unencrypted data are never sent to AWS
    - If the master key is lost the data cannot be decrypted

AWS S3 Best Practices

Performance

- Multiple Concurrent PUTs/GETs
    - S3 scales to support very high request rates. If the request rate grows steadily, S3 automatically partitions the buckets as needed to support higher request rates.
    - If the typical workload involves only occasional bursts of 100 requests per second and less than 800 requests per second, AWS scales and handle it.
    - If the typical workload involves request rate for a bucket to more than 300 PUT/LIST/DELETE requests per second or more than 800 GET requests per second, its recommended to open a support case to prepare for the workload and avoid any temporary limits on your request rate.
    - S3 best practice guidelines can be applied only if you are routinely processing 100 or more requests per second
    - **Workloads that include a mix of request types**
        - If the request workload are typically a mix of GET, PUT, DELETE, or GET Bucket (list objects), choosing appropriate key names for the objects ensures better performance by providing low-latency access to the S3 index
        - This behaviour is driven by how S3 stores key names.
            - S3 maintains an index of object key names in each AWS region.

        - Object keys are stored lexicographically (UTF-8 binary ordering) across multiple partitions in the index i.e. S3 stores key names in alphabetical order.
        - Object keys are stored in across multiple partitions in the index and the key name dictates which partition the key is stored in
        - Using a sequential prefix, such as timestamp or an alphabetical sequence, increases the likelihood that S3 will target a specific partition for a large number of keys, overwhelming the I/O capacity of the partition.
- I**ntroduce some randomness in the key name prefixes, the key names, and the I/O load, will be distributed across multiple index partitions.**
- It also ensures scalability regardless of the number of requests sent per second.

- **Workloads that are GET-intensive**
    - **Cloudfront can be used for performance optimization and can help by:**
        - Distributing content with low latency and high data transfer rate.
        - Caching the content and thereby reducing the number of direct requests to S3
        - Providing multiple endpoints (Edge locations) for data availability
        - Available in two flavours as Web distribution or RTMP distribution

PUTs/GETs for Large Objects

- AWS allows Parallelizing the PUTs/GETs request to improve the upload and download performance as well as the ability to recover in case it fails
- For PUTs, Multipart upload can help improve the uploads by
    - Performing multiple uploads at the same time and maximizing network bandwidth utilization
    - Quick recovery from failures, as only the part that failed to upload needs to be re-uploaded
    - Ability to pause and resume uploads
    - Gegin an upload before the Object size is known
- For GETs, range http header can help to improve the downloads by
    - Allowing the object to be retrieved in parts instead of the whole object
    - Quick recovery from failures, as only the part that failed to download needs to be retried.

List Operations

- Object key names are stored lexicographically in Amazon S3 indexes, making it hard to sort and manipulate the contents of LIST
- S3 maintains a single lexicographically sorted list of indexes

- **Build and maintain Secondary Index outside of S3 for e.g. DynamoDB or RDS to store, index and query objects metadata rather then performing operations on S3**

Security

- Use Versioning
  - It can be used to protect from unintended overwrites and deletions
  - Allows the ability to retrieve and restore deleted objects or roll-back to previous versions
- Enable additional security by configuring a bucket to enable MFA (Multi-Factor Authentication) delete
- Versioning does not prevent Bucket deletion and must be backed up, as if accidentally or maliciously deleted the data is lost
- Use Cross Region replication feature to backup data to a different region
- When using VPC with S3, use VPC S3 endpoints as:
  - Are horizontally scaled, redundant, and highly available VPC components
  - Help establish a private connection between VPC and S3 and the traffic never leaves the Amazon network

Cost

- Optimize S3 storage cost by selecting an appropriate storage class for objects
- Configure appropriate life-cycle management rules to move objects to different storage classes and expire them

Tracking

- Use Event Notifications to be notified for any put or delete request on the S3 objects
- Use CloudTrail, which helps capture specific API calls made to S3 from the AWS account and delivers the log files to an S3 bucket
- Use CloudWatch to monitor the Amazon S3 buckets, tracking metrics such as object counts and bytes stored and configure appropriate actions

• Amazon Glacier

- GLACIER storage class uses the very low-cost Amazon Glacier storage service, but the objects in this storage class are still managed through S3
- GLACIER storage class is suitable for archiving data where data access is infrequent and retrieval time of several (3-5) hours is acceptable.
- Designed for durability of 99.999999999% of objects
- GLACIER cannot be specified as the storage class at the object creation time but has to be transitioned from STANDARD, RRS, or STANDARD_IA to GLACIER storage class using life-cycle management.

When you make a request to retrieve data from Glacier, you initiate a retrieval job for an archive. Once the retrieval job completes, your data will be available to download or access it using Amazon Elastic Compute Cloud (Amazon EC2) for 24 hours. There are three options for retrieving data with varying access times and cost:

- **Expedited retrieval (more expensive):**
  Expedited retrievals allow you to quickly access your data when occasional urgent requests for a subset of archives are required. For all but the largest archives (250MB+), data accessed using Expedited retrievals are typically made available within 1 – 5 minutes. There are two types of Expedited retrievals: On-Demand and Provisioned. On-Demand requests are like EC2 On-Demand instances and are available the vast majority of the time. Provisioned requests are guaranteed to be available when you need them.
- **Standard retrievals:**
  Standard retrievals allow you to access any of your archives within several hours. Standard retrievals typically complete within 3 – 5 hours.
- **Bulk retrievals:**
  Bulk retrievals are Glacier's lowest-cost retrieval option, enabling you to retrieve large amounts, even petabytes, of data inexpensively in a day. Bulk retrievals typically complete within 5 – 12 hours.

Automatically encrypts data using AES-256 (same S3), all data in the service will be encrypted on the server side.

S3 Storage Classes Overview (Important to know)

- S3 storage classes allows life-cycle management for automatic migration of objects for cost savings
- There are three S3 storage classes:
  - **S3 Standard** (it is the default storage class):
- Amazon S3 Standard offers high durability, availability, and performance object storage for frequently accessed data.
- 99.99% availability and 99.999999999% durability
- Supports SSL for data in transit and encryption of data at rest
- Use at least two AZs

- Use cases including cloud applications, dynamic websites, content distribution, mobile and gaming applications, and Big Data analytics
    - **S3 STANDARD_IA** (Infrequent Access) storage class:
        - STANDARD_IA objects are available for real-time access.
        - STANDARD_IA storage class is suitable for larger objects greater than 128 KB (smaller objects are charged for 128KB only) kept for at least 30 days.
        - Same low latency and high throughput performance of Standard
            - **S3 Standard-Infrequent Access**:
- Amazon S3 Standard-Infrequent Access (S3 Standard-IA) is an Amazon S3 storage class for data that is accessed less frequently, but requires rapid access when needed. S3
- Ideal for long-term storage, backups, and as a data store for disaster recovery and older data where access is limited, but the use case still demands high performance
- Use at least two AZs
- 99.9% availability and 99.999999999% durability
- Supports SSL for data in transit and encryption of data at rest
            - **S3 Standard Infrequent Access ONEZONE** (less expensive):
- 99.5% availability and 99.999999999% durability
- Not resilient to the loss of the Availability Zone.
        - **S3 RRS** (Reduced Redundancy Storage):
            - It is designed for non-critical, reproducible data stored at lower levels of redundancy than the STANDARD storage class, which reduces storage costs
            - Cheapest. ideal for objects that data loss is not important. (It is recommended that you not use this storage class. The STANDARD storage class is more cost effective.)
        - 99.99% availability and 99.99% durability
            - RRS stores objects on multiple devices across multiple facilities
            - If an RRS object is lost, S3 returns a 405 error on requests made to that object
            - S3 can send an event notification, configured on the bucket, to alert a user or start a workflow when it detects that an RRS object is lost which can be used to replace the lost object

Life-cycle Policy for an S3 Bucket (Important to know)

-
- A life-cycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:
- You can use it do delete data using the expiration time (for example when uploading logs to a bucket).
- It can be configured using an XML file or through console interface (web):

- Examples:
    - Transition to Standard-IA after, and then type the number of days (for example, 30 days).
    - Transition to One Zone-IA after, and then type the number of days (for example, 30 days).
    - Transition to Amazon Glacier after, and then type the number of days (for example 100 days).
    - Amazon S3 stores the archived objects in Amazon Glacier. However, these are Amazon S3 objects, and you can access them only by using the Amazon S3 console or the Amazon S3 API.

AWS S3 Data Protection (Encryption) – (Important to know)

- Objects are redundantly stored on multiple devices across multiple facilities in an S3 region.
- Amazon S3 PUT and PUT Object copy operations synchronously store the data across multiple facilities before returning SUCCESS.
- Once the objects are stored, S3 maintains its durability by quickly detecting and repairing any lost redundancy.
- S3 also regularly verifies the integrity of data stored using checksums. If Amazon S3 detects data corruption, it is repaired using redundant data.
- In addition, S3 calculates checksums on all network traffic to detect corruption of data packets when storing or retrieving data
- Data protection against accidental overwrites and deletions can be added by enabling Versioning to preserve, retrieve and restore every version of the object stored
- S3 also provides the ability to protect data in-transit (as it travels to and from S3) and at rest (while it is stored in S3)
- **Data Protection**
- **Data in-transit**
    - S3 allows protection of data in-transit by enabling communication via SSL or using client-side encryption

- **Data at Rest**
    - S3 supports both client side encryption and server side encryption for protecting data at rest
    - Using Server-Side Encryption, S3 encrypts the object before saving it on disks in its data centers and decrypt it when the objects are downloaded
    - Using Client-Side Encryption, you can encrypt data client-side and upload the encrypted data to S3. In this case, you manage the encryption process, the encryption keys, and related tools.

- **Server-Side Encryption**
    - Server-side encryption is about data encryption at rest
    - Server-side encryption encrypts only the object data. Any object metadata is not encrypted.

- S3 handles the encryption (as it writes to disks) and decryption (when you access the objects) of the data objects
- There is no difference in the access mechanism for both encrypted or unencrypted objects and is handled transparently by S3

- **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)**
    - Each object is encrypted with a unique data key employing strong multi-factor encryption.
    - SSE-S3 encrypts the data key with a master key that is regularly rotated.
    - S3 server-side encryption uses one of the strongest block ciphers available , 256-bit Advanced Encryption Standard (AES-256), to encrypt the data.
    - Whether or not objects are encrypted with SSE-S3 can’t be enforced when they are uploaded using pre-signed URLs, because the only way you can specify server-side encryption is through the AWS Management Console or through an HTTP request header

- **Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)**
    - SSE-KMS is similar to SSE-S3, but it uses **AWS Key management Services (KMS)** which provides additional benefits along with additional charges
        - KMS is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud.
        - KMS uses customer master keys (CMKs) to encrypt the S3 objects.
        - Master key is never made available
        - KMS enables you to centrally create encryption keys, define the policies that control how keys can be used
        - Allows audit use of key usage to prove they are being used correctly, by inspecting logs in AWS CloudTrail
        - Allows keys to temporarily disabled and re-enabled
        - Allows keys to be rotated regularly
        - Security controls in AWS KMS can help meet encryption-related compliance requirements.
    - SSE-KMS enables separate permissions for the use of an envelope key (that is, a key that protects the data’s encryption key) that provides added protection against unauthorized access of the objects in S3.
    - SSE-KMS provides the option to create and manage encryption keys yourself, or use a default customer master key (CMK) that is unique to you, the service you’re using, and the region you’re working in.
    - Creating and Managing your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.

- Data keys used to encrypt your data are also encrypted and stored alongside the data they protect and are unique to each object
- Process flow
    - An application or AWS service client requests an encryption key to encrypt data and passes a reference to a master key under the account.
    - Client requests are authenticated based on whether they have access to use the master key.
    - A new data encryption key is created, and a copy of it is encrypted under the master key.
    - Both the data key and encrypted data key are returned to the client.
    - Data key is used to encrypt customer data and then deleted as soon as is practical.
    - Encrypted data key is stored for later use and sent back to AWS KMS when the source data needs to be decrypted.

- **Server-Side Encryption usingCustomer-Provided Keys (SSE-C)**
    - Encryption keys can be managed and provided by the Customer and S3 manages the encryption, as it writes to disks, and decryption, when you access the objects
    - When you upload an object, the encryption key is provided as a part of the request and S3 uses that encryption key to apply AES-256 encryption to the data and removes the encryption key from memory.
    - When you download an object, the same encryption key should be provided as a part of the request. S3 first verifies the encryption key and if matches decrypts the object before returning back to you
    - As each object and each object’s version can be encrypted with a different key, you are responsible for maintaining the mapping between the object and the encryption key used.
    - SSE-C request must be done through HTTPS and S3 will reject any requests made over http when using SSE-C.
    - For security considerations, AWS recommends to consider any key sent erroneously using http to be compromised and discarded or rotated
    - S3 does not store the encryption key provided. Instead, it stores a randomly salted HMAC value of the encryption key which can be used to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means, if you lose the encryption key, you lose the object.

- **Client-Side Encryption**

    - Client-side encryption refers to encrypting data before sending it to Amazon S3 and decrypting the data after downloading it

- **AWS KMS-managed customer master key (CMK)**
    - Customer can maintain the encryption CMK with AWS KMS and can provide the CMK id to the client to encrypt the data
    - Uploading Object
        - AWS S3 encryption client first sends a request to AWS KMS for the key to encrypt the object data
        - AWS KMS returns a randomly generated data encryption key with 2 versions a plain text version for encrypting the data and cipher blob to be uploaded with the object as object metadata
        - Client obtains a unique data encryption key for each object it uploads.
        - AWS S3 encryption client uploads the encrypted data and the cipher blob with object metadata
    - Download Object
        - AWS Client first downloads the encrypted object from Amazon S3 along with the cipher blob version of the data encryption key stored as object metadata.
        - AWS Client then sends the cipher blob to AWS KMS to get the plain text version of the same, so that it can decrypt the object data.

- **Client-Side master key**
    - Encryption master keys are completely maintained at Client-side
        - Uploading Object
            - Amazon S3 encryption client (for e.g. AmazonS3EncryptionClient in the AWS SDK for Java) locally generates randomly a one-time-use symmetric key (also known as a data encryption key or data key).
            - Client encrypts the data encryption key using the customer provided master key
            - Client uses this data-encryption key to encrypt the data of a single S3 object (for each object, the client generates a separate data key).
            - Client then uploads the encrypted data to Amazon S3 and also saves the encrypted data key and itsmaterial description as object metadata (x-amz-meta-x-amz-key) in Amazon S3 by default
        - Downloading Object
            - Client first downloads the encrypted object from Amazon S3 along with the object metadata.
            - Using the material description in the metadata, the client first determines which master key to use to decrypt the encrypted data key.
            - Using that master key, the client decrypts the data key and uses it to decrypt the object

- Client-side master keys and your unencrypted data are never sent to AWS
- If the master key is lost the data cannot be decrypted

AWS S3 Best Practices

-
- **Performance**

- Multiple Concurrent PUTs/GETs
  - S3 scales to support very high request rates. If the request rate grows steadily, S3 automatically partitions the buckets as needed to support higher request rates.
  - If the typical workload involves only occasional bursts of 100 requests per second and less than 800 requests per second, AWS scales and handle it.
  - If the typical workload involves request rate for a bucket to more than 300 PUT/LIST/DELETE requests per second or more than 800 GET requests per second, its recommended to open a support case to prepare for the workload and avoid any temporary limits on your request rate.
  - S3 best practice guidelines can be applied only if you are routinely processing 100 or more requests per second
  - **Workloads that include a mix of request types**
    - If the request workload are typically a mix of GET, PUT, DELETE, or GET Bucket (list objects), choosing appropriate key names for the objects ensures better performance by providing low-latency access to the S3 index
    - This behaviour is driven by how S3 stores key names.
    - S3 maintains an index of object key names in each AWS region.
    - Object keys are stored lexicographically (UTF-8 binary ordering) across multiple partitions in the index i.e. S3 stores key names in alphabetical order.
    - Object keys are stored in across multiple partitions in the index and the key name dictates which partition the key is stored in
    - Using a sequential prefix, such as timestamp or an alphabetical sequence, increases the likelihood that S3 will target a specific partition for a large number of keys, overwhelming the I/O capacity of the partition.
- I**ntroduce some randomness in the key name prefixes, the key names, and the I/O load, will be distributed across multiple index partitions.**
- It also ensures scalability regardless of the number of requests sent per second.
-
- **Workloads that are GET-intensive**
- **Cloudfront can be used for performance optimization and can help by:**
- Distributing content with low latency and high data transfer rate.

- Caching the content and thereby reducing the number of direct requests to S3
- Providing multiple endpoints (Edge locations) for data availability
- Available in two flavours as Web distribution or RTMP distribution
-
- **PUTs/GETs for Large Objects**
-
- AWS allows Parallelizing the PUTs/GETs request to improve the upload and download performance as well as the ability to recover in case it fails
- For PUTs, Multipart upload can help improve the uploads by
    - Performing multiple uploads at the same time and maximizing network bandwidth utilization
    - Quick recovery from failures, as only the part that failed to upload needs to be re-uploaded
    - Ability to pause and resume uploads
    - Gegin an upload before the Object size is known
- For GETs, range http header can help to improve the downloads by
    - Allowing the object to be retrieved in parts instead of the whole object
    - Quick recovery from failures, as only the part that failed to download needs to be retried.
- **List Operations**
-
- Object key names are stored lexicographically in Amazon S3 indexes, making it hard to sort and manipulate the contents of LIST
- S3 maintains a single lexicographically sorted list of indexes
- **Build and maintain Secondary Index outside of S3 for e.g. DynamoDB or RDS to store, index and query objects metadata rather then performing operations on S3**
- **Security**
-
- Use Versioning
    - It can be used to protect from unintended overwrites and deletions
    - Allows the ability to retrieve and restore deleted objects or roll-back to previous versions
- Enable additional security by configuring a bucket to enable MFA (Multi-Factor Authentication) delete
- Versioning does not prevent Bucket deletion and must be backed up, as if accidentally or maliciously deleted the data is lost
- Use Cross Region replication feature to backup data to a different region
- When using VPC with S3, use VPC S3 endpoints as:
    - Are horizontally scaled, redundant, and highly available VPC components
    - Help establish a private connection between VPC and S3 and the traffic never leaves the Amazon network
- **Cost**
- Optimize S3 storage cost by selecting an appropriate storage class for objects
- Configure appropriate life-cycle management rules to move objects to different storage classes and expire them

- **Tracking**
- Use Event Notifications to be notified for any put or delete request on the S3 objects
- Use CloudTrail, which helps capture specific API calls made to S3 from the AWS account and delivers the log files to an S3 bucket
- Use CloudWatch to monitor the Amazon S3 buckets, tracking metrics such as object counts and bytes stored and configure appropriate actions

- **S3 - Key points**

- Unless the customer who is uploading the data specifies otherwise, only that customer can access the data, amazon EMR customers can also choose to send data to Amazon S3 using the HTTPS protocol for secure transmission
- Server-side encryption and client-side encryption
- AWS S3 is an object storage, everything saved in S3 is stored as data objects.
  - Each object consists of a MetaData (created by Amazo) and Data (custom data).
  - An object size could be from 0 to 5 TB.
  - S3 Objects are replicated across multiple devices within a region!
  - S3 Objects are saved in a container called “Bucket“, consider Bucket as the root folder.
  - **To prevent accidental object deletion, enable versioning and MFA.**
  - S3 data can be replicated across other regions, this is usually done for compliance. Note: Only new objects will be replicated.
  - Bucket names are unique across all AWS Accounts!
  - Bucket name must be between 3 and 63 characters and can contain numbers, hyphens or periods.
  - Consistency Model of S3
    - When you create a new object, you will receive the latest object. (Read After Write Consistency) – PUTS to new object
    - When you PUT or DELETE a current object, AWS provides Eventual consistency, it might take a while for the changes to be affected.
  - If you triggered an S3 API call and got HTTP 200 result code and MD5 checksum, then it is considered as a successful upload. The S3 API will return an error code in case the upload is unsuccessful.
  - By default, all Amazon S3 resources such as buckets, objects, and related subresources are private which means that only the AWS account holder (resource owner) that created it has access to the resource. The resource owner can optionally grant access permissions to others by writing an access policy. In S3, you also set the permissions of the object during upload to make it public.

Data migrationAWS Snowball.

- Import/export data between on-premises data storage and S3.
- Used for huge data migrationAWS Import/Export Disk
- Transfer data directly to AWS using portable storage devices.

Storage Gateways

AWS Storage Gateway is a hybrid storage service that connects your on-premises (local) applications to seamlessly use AWS cloud storage.

You can use the service for backup and archiving, disaster recovery, cloud data processing, storage tiering, and migration. Your applications connect to the service through a gateway appliance using standard storage protocols, such as NFS, SMB and iSCSI. The gateway connects to AWS storage services, such as Amazon S3, Amazon Glacier, and Amazon EBS, providing storage for files, volumes, and virtual tapes in AWS. The service includes a highly-optimized data transfer mechanism, with bandwidth management, automated network resilience, and efficient data transfer, along with a local cache for low-latency on-premises access to your most active data.

- **AWS Storage Gateway backs up the data in Amazon Storage as incremental EBS snapshots**
- **AWS Storage Gateway, by default, uploads data using SSL and provides data encryption at rest when stored in S3 or Glacier using AES-256**
- **AWS Storage Gateway performs compression of data in-transit and at-rest**

You can run Storage Gateway either on-premises (local data center) as a VM appliance, or in AWS as an Amazon EC2 instance. If your data center goes offline and you don't have an available host, you can deploy a gateway on an EC2 instance. Storage Gateway provides an Amazon Machine Image (AMI) that contains the gateway VM image. Additionally, as you configure a host to deploy a gateway software appliance, you need to allocate sufficient storage for the gateway VM.

- For a gateway deployed on-premises, you chose the type of host, VMware ESXi Hypervisor or Microsoft Hyper-V. and set it up. (Make sure that ports are accessible to the gateway VM).
- For a tape gateway, you have installed client backup software.

AWS Storage Gateway offers file-based, volume-based, and tape-based storage solutions:

- ## File Gateway

The File Gateway configuration offers on-premises applications access to a network share via SMB or NFS connections . File data is cached on the File Gateway appliance for local performance and converted to objects stored in Amazon S3. Data can be stored and retrieved in S3 using NFS (V3 or V4.1) or SMB (V2 or V3) protocols. The software appliance, or gateway, is deployed into your on-premises environment as a VM running on VMware ESXi or Microsoft Hyper-V hypervisor. The gateway provides access to objects in S3 as files or file share mount points.
You can think of a file gateway as an NFS or SMB mount on S3.

- Encryption using AWS Key Management Service (AWS KMS)
- Monitoring using Amazon CloudWatch (CloudWatch)
- Audit using AWS CloudTrail (CloudTrail)
- You can access your data directly in Amazon S3 from any AWS Cloud application or service.

- You can manage your Amazon S3 data using lifecycle policies, cross-region replication, and versioning. You can think of a file gateway as a file system mount on S3.
- Common access management using AWS Identity and Access Management (IAM)

• Volume Gateway

Using volume gateways, you can create storage volumes in the AWS Cloud. Your on-premises applications can access these as Internet Small Computer System Interface (iSCSI) targets. There are two options—cached and stored volumes.
The Volume Gateway configuration connects to on-premises (local) servers and applications as a local disk using iSCSI(Internet Small Computer System Interface) devices. Data in these volumes can be transferred into Amazon S3 cloud storage and accessed through the Volume Gateway. Store data locally for the highest performance (with snapshot backups to the cloud), or blend latency and scale by storing frequently-accessed data locally with "cooler" data in the cloud (with snapshots and clones for protection as well. . T**he gateway supports the following volume configurations:**

- **Cached volumes**

You store your data in Amazon S3 and retain a copy of frequently accessed data subsets locally. Cached volumes offer a substantial cost savings on primary storage and minimize the need to scale your storage on-premises. This approach enables low-latency access to your frequently accessed dataset. By using cached volumes, you can scale your storage resource without having to provision additional hardware (no need to scale your storage on-premises).
Gateway VM can be allocate disks

- **Cache storage**
  Stores the data before uploading it to Amazon S3
- Upload buffer
  Upload buffer acts as a staging area, before the data is uploaded to S3

Gateway uploads data over an encrypted Secure Sockets Layer (SSL) connection to AWS, where it is stored encrypted in Amazon S3

-

• Stored Volumes

With stored volumes, you store the entire set of volume data on-premises (o provide low latency access) and store periodic point-in-time backups (snapshots) in AWS. In this model, your on-premises storage is primary, delivering low-latency access to your entire dataset.
If you need low-latency access to your entire dataset, first configure your on-premises gateway to store all your data locally. Then asynchronously back up point-in-time snapshots of this data to Amazon S3.
This configuration provides durable and inexpensive offsite backups that you can recover to your local data center or Amazon EC2. For example, if you need

replacement capacity for disaster recovery, you can recover the backups to Amazon EC2.

Gateway VM can be allocate disks

- **Volume Storage**
  For storing the actual data
- **Upload buffer**
  Upload buffer acts as a staging area, before the data is uploaded to S3

Gateway uploads data over an encrypted Secure Sockets Layer (SSL) connection to AWS, where it is stored encrypted in Amazon S3

• Tape Gateway (Virtual Tape Library (VTL))

The Tape Gateway configuration replaces backup tapes and tape automation equipment with local disk and cloud storage. Your existing backup and recovery software writes native backup jobs to virtual tapes stored on the Tape Gateway. Virtual tapes can be migrated into Amazon S3 and eventually archived into Amazon Glacier for the lowest cost. Data is accessed through your backup application and the backup catalog maintains complete visibility for all backup jobs and tapes.
With its virtual tape library (VTL) interface, you can use your existing tape-based backup software infrastructure to store data on virtual tape cartridges that you create
You can Cost-effectively and durably archive backup data in Amazon Glacier.
A tape gateway provides a virtual tape infrastructure
Gateway VTL has the following components :

- **Virtual tape**
- **Virtual tape library**
- **Virtual tape shelf**

Questions:
You're running an application on-premises due to its dependency on non-x86 hardware and want to use AWS for data backup. Your backup application is only able to write to POSIX-compatible block-based storage. You have 140TB of data and would like to mount it as a single folder on your file server. Users must be able to access portions of this data while the backups are taking place. What backup solution would be most appropriate for this use case?
Use Storage Gateway and configure it to use Gateway Stored volumes (Data is hosted on the On-premise server as well. The requirement for 140TB is for file server On-Premise more to confuse and not in AWS. Just need a backup solution hence stored instead of cached volumes)
Customer has a single 3-TB volume on-premises that is used to hold a large repository of images and print layout files. This repository is growing at 500 GB a year and must be presented as a single logical volume. The customer is becoming increasingly constrained with their local storage capacity and wants an off-site backup of this data, while maintaining low-latency access to their frequently accessed data. Which AWS Storage Gateway configuration meets the customer requirements?
Gateway-Cached volumes with snapshots scheduled to Amazon S3

You have a proprietary data store on-premises that must be backed up daily by dumping the data store contents to a single compressed 50GB file and sending the file to AWS. Your SLAs state that any dump file backed up within the past 7 days can be retrieved within 2 hours. Your compliance department has stated that all data must be held indefinitely. The time required to restore the data store from a backup is approximately 1 hour. Your on-premise network connection is capable of sustaining 1gbps to AWS. Which backup methods to AWS would be most cost-effective while still meeting all of your requirements?
Transfer the daily backup files to S3 and use appropriate bucket lifecycle policies to send to Glacier (Store in S3 for seven days and then archive to Glacier)
Customer implemented AWS Storage Gateway with a gateway-cached volume at their main office. An event takes the link between the main and branch office offline. Which methods will enable the branch office to access their data? Choose 3 answers
Launch a new AWS Storage Gateway instance AMI in Amazon EC2, and restore from a gateway snapshot
Create an Amazon EBS volume from a gateway snapshot, and mount it to an Amazon EC2 instance.
Launch an AWS Storage Gateway virtual iSCSI device at the branch office, and restore from a gateway snapshot

- ## Storage Gateways – Key notes

  - AWS Storage Gateway service enables hybrid storage between on-premises environments and the AWS Cloud
  - It brings the security, manageability, durability, and scalability of AWS to existing enterprise environments through native integration with AWS encryption, identity management, monitoring, and storage services. Typical use cases include backup and archiving, disaster recovery, moving data to S3 for in-cloud workloads, and tiered storage.
  - AWS Storage Gateway supports three storage interfaces: file, volume, and tape. Each gateway you have can provide one type of interface.
    - **The *file gateway*** enables you to store and retrieve objects in Amazon S3 using file protocols, such as NFS. Objects written through file gateway can be directly accessed in S3.
    - **The *volume gateway*** provides block storage to your applications using the iSCSI protocol. Data on the volumes is stored in Amazon S3. To access your iSCSI volumes in AWS, you can take EBS snapshots which can be used to create EBS volumes.
    - The volume gateway runs in either a cached or stored mode.
      - In the **cached mode,** you store your data in S3, and retain a copy of frequently accessed data subsets locally in your on-premise network. Cached volumes offer cost savings on primary storage and m**inimize the need to scale your storage on-premises. You**

also retain low-latency access to your frequently accessed data.

- In the **stored mode,** your primary data is stored locally and your entire dataset is available for low-latency access while asynchronously backed up to AWS.

- The ***tape gateway*** provides your backup application with an iSCSI virtual tape library (VTL) interface, consisting of a virtual media changer, virtual tape drives, and virtual tapes. Virtual tape data is stored in Amazon S3 or can be archived to Amazon Glacier.

- **Encryption:** All data transferred between any type of gateway appliance and AWS storage is encrypted using SSL. By default, all data stored by AWS Storage Gateway in S3 is encrypted server-side with Amazon S3-Managed Encryption Keys (SSE-S3). Also, you can optionally configure different gateway types to encrypt stored data with AWS Key Management Service (KMS) via the Storage Gateway. AWS Storage Gateway is HIPAA eligible.

AWS Elastic Load Balancing

- Elastic Load Balancing allows the incoming traffic to be distributed automatically across multiple healthy EC2 instances.
- ELB serves as a single point of contact to the user
- ELB helps to being transparent and increases the application availability by allowing addition or removal of multiple EC2 instances across one or more availability zones, without disrupting the overall flow of information.
- ELB benefits
    - it is itself a distributed system that is fault tolerant and actively monitored
    - Abstracts out the complexity of managing, maintaining, and scaling load balancers
    - It can also serve as the first line of defense against attacks on network.
    - It can offload the work of encryption and decryption (SSL termination)
    - It offers integration with Auto Scaling, which ensures enough back-end capacity available to meet varying traffic levels
    - They are engineered to not be a single point of failure
- Elastic Load Balancer, by default, routes each request independently to the registered instance with the smallest load.
- If an EC2 instance fails, ELB automatically reroutes the traffic to the remaining running healthy EC2 instances. If a failed EC2 instance is restored, Elastic Load Balancing restores the traffic to that instance.
- Load Balancers only work across AZs within a region

- # ELB Key Points

  - **Scaling ELB**
    - Each ELB is allocated and configured with a default capacity
    - ELB Controller is the service which stores all the configuration and also monitors the load balancer and manages the capacity that is used to handle the client requests
    - As the traffic profile changes, the controller service scales the load balancers to handle more requests, scaling equally in all AZs.
    - ELB increases its capacity by utilizing either larger resources (scale up – resources with higher performance characteristics) or more individual resources (scale out).
    - AWS itself handles the scaling of the ELB capacity and this scaling is different to scaling of the EC2 instances to which the ELB routes its request to, which is handled by Auto Scaling
    - Time required for Elastic Load Balancing to scale can range from 1 to 7 minutes, depending on the changes in the traffic profile

  - **Pre-Warming ELB**
    - ELB works best with gradual increase in traffic
    - AWS is able to scale automatically and handle a vast majority of use cases
    - However, in certain scenarios, if there is a flash traffic spike expected or a load test cannot be configured to gradually increase traffic, recommended to contact AWS support to have the load balancer “pre-warmed”
    - AWS will help Pre-warming the ELB, by configuring the load balancer to have the appropriate level of capacity based on expected traffic
    - AWS would need the information for the start, end dates and the expected request rate per second with the total size of request/response.

  - **DNS Resolution**
    - ELB is scaled automatically depending on the traffic profile
    - When scaled, Elastic Load Balancing service will update the Domain Name System (DNS) record of the load balancer so that the new resources have their respective IP addresses registered in DNS.
    - DNS record created includes a Time-to-Live (TTL) setting of 60 seconds
    - By default, ELB will return multiple IP addresses when clients perform a DNS resolution, with the records being randomly ordered on each DNS resolution request.
    - It is recommended that clients will re-lookup the DNS at least every 60 seconds to take advantage of the increased capacity
  - **Load Balancer Types**
    - Internet Load Balancer
      - An Internet-facing load balancer takes requests from clients **over the Internet** and distributes them across the EC2 instances that are registered with the load balancer
    - Internal Load Balancer

        - Internal load balancer routes traffic to EC2 instances in **private subnets**

- **Availability Zones/Subnets**
    - Elastic Load Balancing allows subnets to be added and creates a load balancer node in each of the Availability Zone where the subnet resides.
    - Elastic Load Balancer should have at least one subnet attached
    - Only one subnet per AZ can be attached to the ELB. Attaching a subnet with an AZ already attached replaces the existing subnet
    - Each Subnet must have a CIDR block with at least a /27 bitmask and has at least 8 free IP addresses, which ELB uses to establish connections with the back-end instances.
    - For High Availability, it is recommended to attach one subnet per AZ for at least two AZs, even if the instances are in a single subnet.
    - Subnets can be attached or detached from the ELB and it would start or stop sending requests to the instances in the subnet accordingly

- **Security Groups & NACL**
    - Security groups & NACLs should allow Inbound traffic, on the load balancer listener port, from the Client for an Internet ELB or VPC CIDR for an Internal ELB
    - Security groups & NACLs should allow Outbound traffic to the back-end instances on both the instance listener port and the health check port
    - NACLs, in addition, should allow responses on the ephemeral ports
    - All EC2 instances should allow incoming traffic from ELB
        - Security Groups for Load Balancers in a VPC
          When you use the AWS Management Console to create a load balancer in a VPC, you can choose an existing security group for the VPC or create a new security group for the VPC. If you choose an existing security group, it must allow traffic in both directions to the listener and health check ports for the load balancer. If you choose to create a security group, the console automatically adds rules to allow all traffic on these ports.
          [Nondefault VPC] If you don't specify a security group, your load balancer is automatically associated with the default security group for the VPC.
          [Default VPC] You can't choose an existing security group for your load balancer. Instead, Elastic Load Balancing provides a security group with rules to allow all traffic on the ports specified for the load balancer. and health check ports for the new load balancer. When you delete your load balancer, this security group is not deleted automatically.

Recommended Rules for Load Balancer Security Groups
The security groups for your load balancers must allow them to communicate with your instances. The recommended rules depend on the type of load balancer (Internet-facing or internal).

Internet-facing Load Balancer: Recommended Rules

Inbound			
Source	**Protocol**	**Port Range**	**Comment**
0.0.0.0/0	TCP	listener	Allow all inbound traffic on the load balancer listener port
Outbound			
Destination	**Protocol**	**Port Range**	**Comment**
instance security group	TCP	instance listener	Allow outbound traffic to instances on the instance listener port
instance security group	TCP	health check	Allow outbound traffic to instances on the health check port

Internal Load Balancer: Recommended Rules

Inbound			
Source	**Protocol**	**Port Range**	**Comment**
VPC CIDR	TCP	listener	Allow inbound traffic from the VPC CIDR on the load balancer listener port
Outbound			
Destination	**Protocol**	**Port Range**	**Comment**
instance security group	TCP	instance listener	Allow outbound traffic to instances on the instance listener port
instance security group	TCP	health check	Allow outbound traffic to instances on the health check port

Network ACLs for Load Balancers in a VPC

The default network access control list (ACL) for the VPC allows all inbound and outbound traffic. If you create custom network ACLs, you must add rules that allow the load balancer and instances to communicate.
The recommended rules for the subnet for your load balancer depend on the type of load balancer (Internet-facing or internal).

Internet-Facing Load Balancer: Recommended Rules

Inbound			
Source	**Protocol**	**Port**	**Comment**
0.0.0.0/0	TCP	listener	Allow all inbound traffic on the load balancer listener port
VPC CIDR	TCP	1024-65535	Allow inbound traffic from the VPC CIDR on the ephemeral ports
Outbound			
Destination	**Protocol**	**Port**	**Comment**
VPC CIDR	TCP	instance listener	Allow all outbound traffic on the instance listener port
VPC CIDR	TCP	health check	Allow all outbound traffic on the health check port
0.0.0.0/0	TCP	1024-65535	Allow all outbound traffic on the ephemeral ports

Internal Load Balancer: Recommended Rules

Inbound			
Source	**Protocol**	**Port**	**Comment**
VPC CIDR	TCP	listener	Allow inbound traffic from the VPC CIDR on the load balancer listener port
VPC CIDR	TCP	1024-65535	Allow inbound traffic from the VPC CIDR on the ephemeral ports
Outbound			
Destination	**Protocol**	**Port**	**Comment**
VPC CIDR	TCP	instance listener	Allow outbound traffic to the VPC CIDR on the instance listener port
VPC CIDR	TCP	health check	Allow outbound traffic to the VPC CIDR on the health check port
VPC CIDR	TCP	1024-65535	Allow outbound traffic to the VPC CIDR on the ephemeral ports

- **SSL Negotiation Configuration**
  - For HTTPS load balancer, Elastic Load Balancing uses an Secure Socket Layer (SSL) negotiation configuration, known as a security policy, to negotiate SSL connections between a client and the load balancer.
  - A security policy is a combination of SSL protocols, SSL ciphers, and the Server Order Preference option
    - Elastic Load Balancing supports the following versions of the SSL protocol TLS 1.2, TLS 1.1, TLS 1.0, SSL 3.0, SSL 2.0 (deprecated now)
    - SSL protocols use several SSL ciphers to encrypt data over the Internet.
    - Elastic Load Balancing supports the Server Order Preference option for negotiating connections between a client and a load balancer.

        - During the SSL connection negotiation process, this allows the load balancer to control and select the first cipher in its list that is in the client's list of ciphers instead of the default behaviour of checking matching first cipher in client's list with server's list.
    - Elastic Load Balancer allows using a Predefined Security Policies or creating a Custom Security Policy for specific needs. **If none is specified, ELB selects the latest Predefined Security Policy.**

- **Health Checks**
    - Load balancer performs health checks on all registered instances, whether the instance is in a healthy state or an unhealthy state.
    - Load balancer performs health checks to discover the availability of the EC2 instances, the load balancer periodically sends pings, attempts connections, or sends request to health check the EC2 instances.
    - Health check is **InService** for status of healthy instances and **OutOfService** for unhealthy ones
    - Load balancer sends a request to each registered instance at the Ping Protocol, Ping Port and Ping Path every HealthCheck Interval seconds. It waits for the instance to respond within the Response Timeout period. If the health checks exceed the Unhealthy Threshold for consecutive failed responses, the load balancer takes the instance out of service. When the health checks exceed the Healthy Threshold for consecutive successful responses, the load balancer puts the instance back in service.
    - Load balancer only sends requests to the healthy EC2 instances and stops routing requests to the unhealthy instances

- **Listeners**
    - Listeners is the process which checks for connection requests from client
    - Listeners are configured with a protocol and a port for front-end (client to load balancer) connections, and a protocol and a port for back-end (load balancer to back-end instance) connections.
    - Listeners support HTTP, HTTPS, SSL, TCP protocols
    - A X.509 certificate is required for HTTPS or SSL connections and load balancer uses the certificate to terminate the connection and then decrypt requests from clients before sending them to the back-end instances.
    - If you want to use SSL, but don't want to terminate the connection on the load balancer, use TCP for connections from the client to the load balancer, use the SSL protocol for connections from the load balancer to the back-end application, and deploy certificates on the back-end instances handling requests.
    - If you use an HTTPS/SSL connection for your back end, you can enable authentication on the back-end instance. This authentication can be used to ensure that back-end instances accept only

encrypted communication, and to ensure that the back-end instance has the correct certificates.
    - ELB HTTPS listener does not support Client-Side SSL certificates

- **Idle Connection Timeout**
    - **For each request that a client makes through a load balancer, it maintains two connections, for each client request, one connection with the client and the other connection is to the back-end instance.**
    - For each connection, the load balancer manages an idle time-out that is triggered when no data is sent over the connection for a specified time period. If no data has been sent or received, it closes the connection after the idle time-out period (defaults to 60 seconds) has elapsed
    - For lengthy operations, such as file uploads, the idle time-out setting for the connections should be adjusted to ensure that lengthy operations have time to complete.

- **X-Forwarded Headers & Proxy Protocol Support**
    - As the Elastic Load Balancer intercepts the traffic between the client and the back end servers, the back end server does not know the IP address, Protocol and the Port used between the Client and the Load balancer.
    - ELB provides X-Forwarded headers support to help back end servers track the same when using HTTP protocol
        - X-Forwarded-For request header to help back end servers identify the IP address of a client when you use an HTTP or HTTPS load balancer.
        - X-Forwarded-Proto request header to help back end servers identify the protocol (HTTP/S) that a client used to connect to the server
        - X-Forwarded-Port request header to help back end servers identify the port that an HTTP or HTTPS load balancer uses to connect to the client.
    - ELB provides Proxy Protocol support to help back end servers track the same when using non-HTTP protocol or when using HTTPS and not terminating the SSL connection on the load balancer.
        - Proxy Protocol is an Internet protocol used to carry connection information from the source requesting the connection to the destination for which the connection was requested.
        - Elastic Load Balancing uses Proxy Protocol version 1, which uses a human-readable header format with connection information such as the source IP address, destination IP address, and port numbers
        - If the ELB is already behind a Proxy with the Proxy protocol enabled, enabling the Proxy Protocol on ELB would add the header twice

- **Cross-Zone**
    - By default, the load balancer distributes incoming requests evenly across its enabled Availability Zones for e.g. If AZ-a has 5 instances

and AZ-b has 2 instances, the load will still be distributed 50% across each of the AZs
    - Enabling Cross-Zone load balancing allows the ELB to distribute incoming requests evenly across all the back-end instances, regardless of the AZ
    - Cross-zone load balancer reduces the need to maintain equivalent numbers of back-end instances in each Availability Zone, and improves application's ability to handle the loss of one or more back-end instances.
    - It is still recommended to maintain approximately equivalent numbers of instances in each Availability Zone for higher fault tolerance.

- **Connection Draining**
    - By default, if an registered EC2 instance with the ELB is de registered or becomes unhealthy, the load balancer immediately closes the connection
    - Connection draining can help the load balancer to complete the in-flight requests made while keeping the existing connections open, and preventing any new requests being sent to the instances that are de-registering or unhealthy.
    - Connection draining helps perform maintenance such as deploying software upgrades or replacing back-end instances without affecting customers' experience
    - Connection draining allows you to specify a maximum time (between 1 and 3,600 seconds and default 300 seconds) to keep the connections alive before reporting the instance as de-registered. Maximum time-out limit does not apply to connections to unhealthy instances.
    - If the instances are part of an Auto Scaling group and connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete, or for the maximum time-out to expire, before terminating instances due to a scaling event or health check replacement.

- **Sticky Sessions (Session Affinity)**
    - ELB can be configured to use **sticky session** feature (also called session affinity) **which enables it to bind a user's session to an instance and ensures all requests are sent to the same instance.**
    - Stickiness remains for a period of time which can be controlled by the application's session cookie, if one exists, or through cookie, named AWS ELB, created through Elastic Load balancer.
    - **Sticky sessions for ELB are disabled, by default.**

- **Requirements**
    - An HTTP/HTTPS load balancer.
    - SSL traffic should be terminated on the ELB. ELB does session stickiness on a HTTP/HTTPS listener is by utilizing an HTTP cookie. If SSL traffic is not terminated on the ELB and is terminated on the back-end instance, the ELB has no visibility into the HTTP headers

and therefore can not set or read any of the HTTP headers being passed back and forth.
    - At least one healthy instance in each Availability Zone.

- **Duration-Based Session Stickiness**
    - Duration-Based Session Stickiness is maintained by ELB using a special cookie created to track the instance for each request to each listener.
    - When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the request is sent to the instance specified in the cookie.
    - If there is no cookie, the ELB chooses an instance based on the existing load balancing algorithm and a cookie is inserted into the response for binding subsequent requests from the same user to that instance.
    - Stickiness policy configuration defines a cookie expiration, which establishes the duration of validity for each cookie. The cookie is automatically updated after its duration expires.

- **Application-Controlled Session Stickiness**
    - Load balancer uses a special cookie only to associate the session with the instance that handled the initial request, but follows the lifetime of the application cookie specified in the policy configuration.
    - Load balancer only inserts a new stickiness cookie if the application response includes a new application cookie. The load balancer stickiness cookie does not update with each request.
    - If the application cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.
    - If an instance fails or becomes unhealthy, the load balancer stops routing request to that instance and instead of, it chooses a new healthy instance based on the existing load balancing algorithm. The load balancer treats the session as now “stuck” to the new healthy instance, and continues routing requests to that instance even if the failed instance comes back.

- **Load Balancer Deletion**
    - Deleting a load balancer does not affect the instances registered with the load balancer and they would continue to run

ELB with Autoscaling

- Auto Scaling dynamically adds and removes EC2 instances, while Elastic Load Balancing manages incoming requests by optimally routing traffic so that no one instance is overwhelmed
- Auto Scaling helps to automatically increase the number of EC2 instances when the user demand goes up, and decrease the number of EC2 instances when demand goes down
- ELB service helps to distribute the incoming web traffic (called the load) automatically among all the running EC2 instances

- ELB uses load balancers to monitor traffic and handle requests that come through the Internet.
- Using ELB & Auto Scaling
    - Makes it easy to route traffic across a dynamically changing fleet of EC2 instances
    - Load balancer acts as a single point of contact for all incoming traffic to the instances in an Auto Scaling group.

Attaching/Detaching ELB with Auto Scaling Group

- Auto Scaling integrates with Elastic Load Balancing and enables to attach one or more load balancers to an existing Auto Scaling group.
- ELB registers the EC2 instance using its IP address and routes requests to the primary IP address of the primary interface (eth0) of the instance.
- After the ELB is attached, it automatically registers the instances in the group and distributes incoming traffic across the instances
- When ELB is detached, it enters the Removing state while de-registering the instances in the group.
- If connection draining is enabled, ELB waits for in-flight requests to complete before de registering the instances.
- Instances remain running after they are de registered from the ELB
- Auto Scaling adds instances to the ELB as they are launched, but this can be suspended. Instances launched during the suspension period are not added to load balancer, after resumption, and must be registered manually.

High Availability & Redundancy

- Auto Scaling can span across multiple AZs, within the same region
- When one AZ becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected AZ
- When the unhealthy AZs recovers, Auto Scaling redistributes the traffic across all the healthy AZs
- Elastic Load balancer can be setup to distribute incoming requests across EC2 instances in a single AZ or multiple AZs within a region
- It is recommended to take advantage of the safety and reliability of geographic redundancy by using Auto Scaling & ELB by spanning Auto Scaling groups across multiple AZs within a region and then setting up ELB to distribute incoming traffic across those AZs.
- Incoming traffic is load balanced equally across all the AZs enabled for ELB

Health Checks

- Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks
- Auto Scaling marks the instance as unhealthy and replaces the instance, if the instance fails the EC2 instance status check
- ELB also performs health checks on the EC2 instances that are registered with the it for e.g. application is available by pinging and health check page
- Auto Scaling, by default, does not replace the instance, if the ELB health check fails

- ELB health check with the instances should be used to ensure that traffic is routed only to the healthy instances
- After a load balancer is registered with an Auto Scaling group, it can be configured to use the results of the ELB health check in addition to the EC2 instance status checks to determine the health of the EC2 instances in the Auto Scaling group.

Monitoring

- Elastic Load Balancing sends data about the load balancers and EC2 instances to Amazon CloudWatch. CloudWatch collects data about the performance of your resources and presents it as metrics.
- After registering one or more load balancers with the Auto Scaling group, Auto Scaling group can be configured to use **ELB metrics** (**such as request latency or request count**) to **scale the application automatically**

AWS Application Load Balancer

- It operates at the layer 7 (application layer) and allows defining routing rules based on content across multiple services or containers running on one or more EC2 instances.

Application Load Balancer Benefits

- Support for **Path-based routing**, where listener rules can be configured to forward requests based on the URL in the request. **This enables structuring application as smaller services, and route requests to the correct service based on the content of the URL.**
- **Support for routing requests to multiple services on a single EC2 instance** by registering the instance using multiple ports.
- **Support for containerized applications**. EC2 Container Service (ECS) can select an unused port when scheduling a task and register the task with a target group using this port, enabling efficient use of the clusters.
- Support for monitoring the health of each service independently, as health checks are defined at the target group level and many CloudWatch metrics are reported at the target group level. **Attaching a target group to an Auto Scaling group enables you to scale each service dynamically based on demand.**

- **Application Load Balancer Features**
  - Supports load balancing of applications using HTTP and HTTPS (Secure HTTP) protocols
  - Supports HTTP/2, which is enabled natively. Clients that support HTTP/2 can connect over TLS
  - Supports WebSockets and Secure WebSockets natively
  - Supports Request tracing, by default

- Supports Sticky Sessions (Session Affinity) using load balancer generated cookies, to route requests from the same client to the same target
- Supports SSL termination, to decrypt the request on ALB before sending it to the underlying targets.
- Supports layer 7 specific features like X-Forwarded-For headers to help determine the actual client IP, port and protocol
- Automatically scales its request handling capacity in response to incoming application traffic.
- Provides High Availability, by allowing you to specify more than one AZ and distribution of incoming traffic across multiple AZs.
- Integrates with ACM to provision and bind a SSL/TLS certificate to the load balancer thereby making the entire SSL offload process very easy
- Supports IPv6 addressing, for an Internet facing load balancer
- Supports Request Tracking, where in a new custom identifier “X-Amzn-Trace-Id” HTTP header is injected on all requests to help track in the request flow across various services
- Supports Security Groups to control the traffic allowed to and from the load balancer.
- Provides Access Logs, to record all requests sent the load balancer, and store the logs in S3 for later analysis in compressed format
- Provides Delete Protection, to prevent the ALB from accidental deletion
- Supports Connection Idle Timeout – ALB maintains two connections for each request one with the Client (front end) and one with the target instance (back end). If no data has been sent or received by the time that the idle time-out period elapses, ALB closes the front-end connection
- Integrates with CloudWatch to provide metrics such as request counts, error counts, error types, and request latency
- Integrates with AWS WAF, a web application firewall that helps protect web applications from attacks by allowing rules configuration based on IP addresses, HTTP headers, and custom URI strings
- Integrates with CloudTrail to receive a history of ALB API calls made on the AWS account

- **Application Load Balancer Listeners**
  - A listener is a process that checks for connection requests, using the configured protocol and port
  - Listener supports HTTP & HTTPS protocol with Ports from 1-65535
  - ALB supports SSL Termination for HTTPS listener, which helps to offload the work of encryption and decryption so that the targets can focus on their main work.
  - HTTPS listener supports exactly one SSL server certificate on the listener.
  - WebSockets with both HTTP and HTTPS listeners (Secure WebSockets)
  - Supports HTTP/2 with HTTPS listeners
  - 128 requests can be sent in parallel using one HTTP/2 connection.
  - ALB converts these to individual HTTP/1.1 requests and distributes them across the healthy targets in the target group using the round robin routing algorithm.

- HTTP/2 uses front-end connections more efficiently resulting in fewer connections between clients and the load balancer.
- Server-push feature of HTTP/2 is not supported
- Each listener has a default rule, and can optionally define additional rules.
- Each rule consists of a priority, action, optional host condition, and optional path condition.
- **Priority** – Rules are evaluated in priority order, from the lowest value to the highest value. The default rule has lowest priority
- **Action** – Each rule action has a type and a target group. Currently, the only supported type is forward, which forwards requests to the target group. You can change the target group for a rule at any time.
- **Condition** – There are two types of rule conditions: host and path. When the conditions for a rule are met, then its action is taken
- Host Condition or Host-based routing
- Host conditions can be used to define rules that forward requests to different target groups based on the host name in the host header
- This enables support for multiple domains using a single ALB for e.g. orders.example.com, images.example.com, registration.example.com
- Each host condition has one hostname. If the hostname in
- Path Condition or path-based routing
- Path conditions can be used to define rules that forward requests to different target groups based on the URL in the request
- Each path condition has one path pattern for e.g. example.com/orders, example.com/images, example.com/registration
- If the URL in a request matches the path pattern in a listener rule exactly, the request is routed using that rule.

Advantages over Classic Load Balancer

- Support for path-based routing, where rules can be configured for the listener to forward requests based on the URL
- Support for host-based routing, where rules can be configured for the listener to forward requests based on the host field in the HTTP header.
- Support for routing requests to multiple applications on a single EC2 instance. Each instance or IP address can be registered with the same target group using multiple ports
- Support for registering targets by IP address, including targets outside the VPC for the load balancer.
- Support containerized applications with ECS using Dynamic port mapping
- Support monitoring the health of each service independently, as health checks and many CloudWatch metrics are defined at the target group level
- Attaching of target group to an Auto Scaling group enables scaling of each service dynamically based on demand
- Access logs contain additional information & stored in compressed format
- Improved load balancer performance.

AWS ELB Network Load Balancer

- **Network Load Balancer operates at the connection level (Layer 4), routing connections to targets – EC2 instances, containers and IP addresses based on IP protocol data.**
- Network Load Balancer is suited for load balancing of TCP traffic
- Network Load Balancer is capable of handling millions of requests per second while maintaining ultra-low latencies.
- Network Load Balancer is optimized to handle sudden and volatile traffic patterns while using a single static IP address per Availability Zone.
- NLB is integrated with other AWS services such as Auto Scaling, EC2 Container Service (ECS), and CloudFormation.

Network Load Balancer Features

Connection-based Load Balancing

- Allows load balancing of TCP traffic, routing connections to targets – EC2 instances, microservices and containers, and IP addresses.

High Availability

- Is highly available.
- Accepts incoming traffic from clients and distributes this traffic across the targets within the same Availability Zone.
- Monitors the health of its registered targets and routes the traffic only to healthy targets
- If a health check fails and an unhealthy target is detected, it stops routing traffic to that target and reroutes traffic to remaining healthy targets.
- If configured with multiple AZs and if all the targets in a single AZ fail, it routes traffic to healthy targets in the other AZs

High Throughput

- Is designed to handle traffic as it grows and can load balance millions of requests/sec.
- Can also handle sudden volatile traffic patterns.

Low Latency

- Offers extremely low latencies for latency-sensitive applications.

Preserve source IP address

- Preserves client side source IP allowing the back-end to see client IP address

Static IP support

- Automatically provides a static IP per Availability Zone (subnet) that can be used by applications as the front-end IP of the load balancer.

Elastic IP support

- An Elastic IP per Availability Zone (subnet) can also be assigned, optionally, thereby providing a fixed IP.

Health Checks

- Supports both network and application target health checks.
- Network-level health check
    - Is based on the overall response of the underlying target (instance or a container) to normal traffic.
    - Target is marked unavailable if it is slow or unable to respond to new connection requests
- Application-level health check
    - Is based on a specific URL on a given target to test the application health deeper

DNS Fail-over

- **Integrates with Route 53**
- Route 53 will direct traffic to load balancer nodes in other AZs, if there are no healthy targets with NLB or if the NLB itself is unhealthy
- If NLB is unresponsive, Route 53 will remove the unavailable load balancer IP address from service and direct traffic to an alternate Network Load Balancer in another region.

Integration with AWS Services

- Is integrated with other AWS services such as Auto Scaling, EC2 Container Service (ECS), CloudFormation, CodeDeploy, and AWS Config.

Long-lived TCP Connections

- Supports long-lived TCP connections ideal for WebSocket type of applications

Central API Support

- Uses the same API as Application Load Balancer.
- **Enables you to work with target groups, health checks, and load balance across multiple ports on the same EC2 instance to support containerized applications.**

Robust Monitoring and Auditing

- Integrated with CloudWatch to report Network Load Balancer metrics.
- CloudWatch provides metrics such as Active Flow count, Healthy Host Count, New Flow Count, Processed bytes, and more.
- integrated with CloudTrail to track API calls to the NLB

Enhanced Logging

- use the Flow Logs feature to record all requests sent to the load balancer.
- Flow Logs capture information about the IP traffic going to and from network interfaces in the VPC
- Flow log data is stored using CloudWatch Logs

Zonal Isolation

- Is designed for application architectures in a single zone.
- Can be enabled in a single AZ to support architectures that require zonal isolation
- Automatically fails-over to other healthy AZs, if something fails in a AZ
- Its recommended to configure the load balancer and targets in multiple AZs for achieving high availability

Load Balancing using IP addresses as Targets

- Allows load balancing of any application hosted in AWS or on-premises using IP addresses of the application backends as targets.
- Allows load balancing to an application backend hosted on any IP address and any interface on an instance.
- Ability to load balance across AWS and on-premises resources helps migrate-to-cloud, burst-to-cloud or failover-to-cloud
- Applications hosted in on-premises locations can be used as targets over a Direct Connect connection and EC2-Classic (using ClassicLink).

Advantages over Classic Load Balancer

- Ability to handle volatile workloads and scale to millions of requests per second, without the need of pre-warming
- Support for static IP/Elastic IP addresses for the load balancer
- Support for registering targets by IP address, including targets outside the VPC (on-premises) for the load balancer.
- Support for routing requests to multiple applications on a single EC2 instance. Single instance or IP address can be registered with the same target group using multiple ports.
- Support for containerized applications. Using Dynamic port mapping, ECS can select an unused port when scheduling a task and register the task with a target group using this port.
- Support for monitoring the health of each service independently, as health checks are defined at the target group level and many CloudWatch metrics are reported at the target group level. Attaching a target group to an Auto Scaling group enables scaling each service dynamically based on demand

AWS Elastic or Classic Load Balancer (ELB) vs Application Load Balancer

Content below lists down the feature comparison for both.

Usage Pattern

- An ELB is ideal for simple load balancing of traffic across multiple EC2 instances,
- Application Load Balancer is ideal for micro services or container-based architectures where there is a need to route traffic to multiple services or load balance across multiple ports on the same EC2 instance. (it supports web-sockets)

MAIN DIFFERENCES

- **ELB or Classic Load Balancer**
  - It operates at layer 4 and supports HTTP, HTTPS, TCP, SSL
  - Supports back-end Server Authentication
    - Back-end Server Authentication enables authentication of the instances. Load balancer communicates with an instance only if the public key that the instance presents to the load balancer matches a public key in the authentication policy for the load balancer.

- **ALB**
  - It operates at layer 7 and supports HTTP, HTTPS, HTTP/2, WebSockets
  - Host-based Routing & Path-based Routing
    - Host-based routing use host conditions to define rules that forward requests to different target groups based on the host name in the host header. **This enables ALB to support multiple domains using a single load balancer**.
    - Path-based routing use path conditions to define rules that forward requests to different target groups based on the URL in the request. Each path condition has one path pattern. If the URL in a request matches the path pattern in a listener rule exactly, the request is routed using that rule.
  - Dynamic Ports
    - Only ALB supports Dynamic Port Mapping with ECS, which allows two containers of a service to run on a single server on dynamic ports that ALB automatically detects and reconfigures itself.
  - Deletion Protection
    - Only ALB supports Deletion Protection
  - Request Tracing
    - Only ALB supports Request Tracing to track HTTP requests from clients to targets or other services.
  - IPv6 in VPC
    - Only ALB supports IPv6 in VPC
  - AWS WAF
    - Only ALB supports AWS WAF, which can be directly used on ALBs (both internal and external) in a VPC, to protect websites and web services

• ELB Key Points to remember

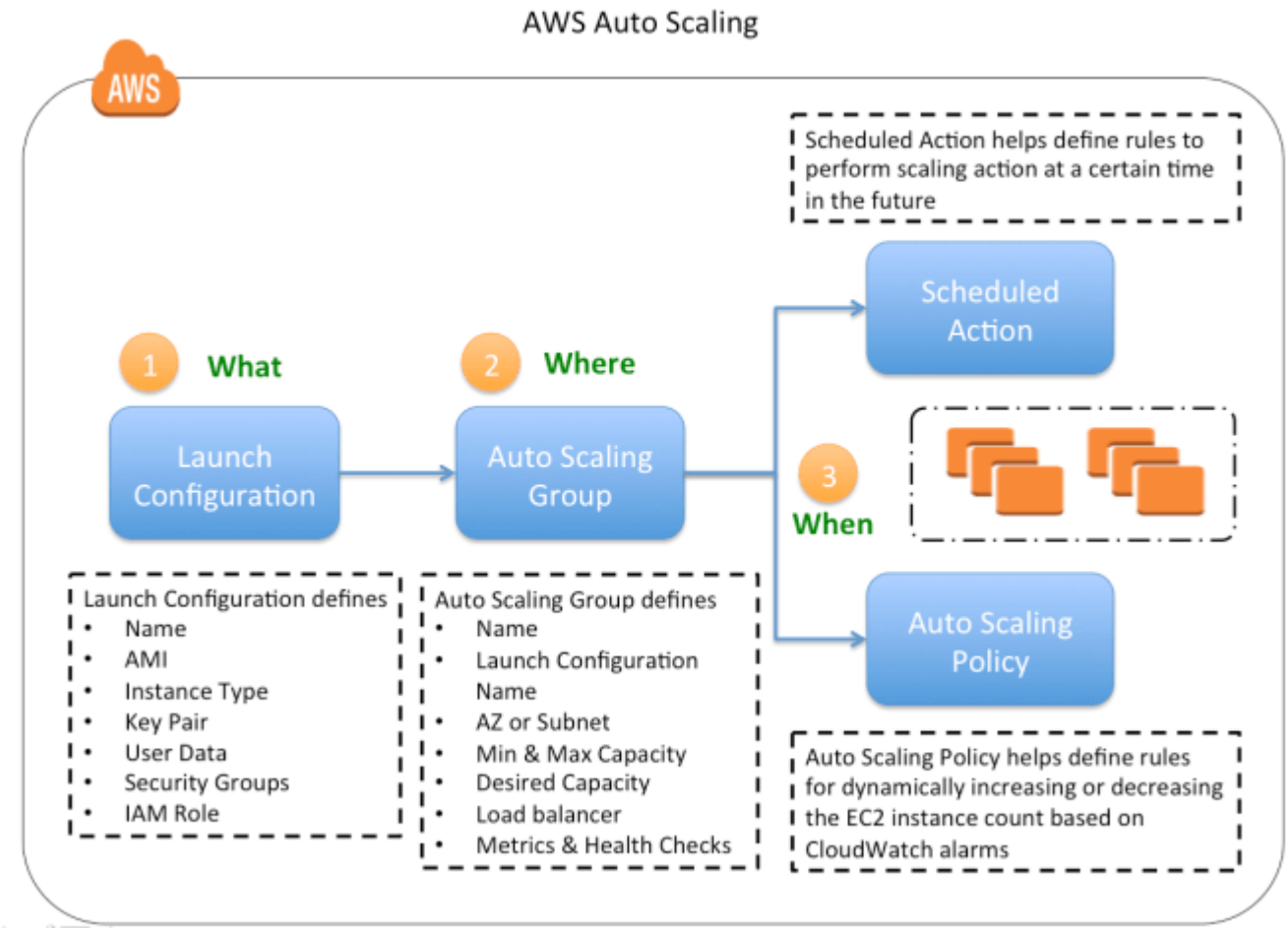
- **There are two types of load balancer:**
  - Internet Facing: has public IP (public subnet)
  - Internal Load Balancer: routes traffic within a VPC) (private subnet)
- **There are three categories of :**
  - **Application Load Balancer**
    - Operates at Layer 7.
    - Supports WebSockets and Secure WebSockets.
    - Supports SNI.
    - Supports IPv6.
  - **Network Load Balancer** (The NLB is the recommended Load Balancer to be used at Layer 4)
    - Operates at Layer 4.
    - Preserves the source IP.
    - Provides reduced latency compared to other Load Balancers.
    - Handles millions of requests per seconds.
  - **Elastic or Classic Load Balancer**
    - Operates at Layer 4.
    - Features Cross Zone Load Balancing.

- Idle Connection Timeout: Is the period of inactivity between the client and the EC2 instance in which the Load Balancer terminates the connection. By default, it is 60 seconds.
- Enable Cross Zone Load Balancing, to distribute traffic evenly across the pool of registered AWS instances.
- Connection Draining stops the load balancer sending traffic to faulty instances.
- Proxy Protocol: To receive the clients IP and User Agent, it needs to be enabled. (only on Network Load Balancer and Elastic Load Balancer)
- If both sides are SSL, does not support proxy protocol, **TCP to TCP it support proxy protocol**, **SSL to TCP support the proxy protocol**, **TCP to SSL does not support proxy protocol**
- Sticky Sessions: Ensures that the load balancer sends future user request to the EC2 instance that received the initial request. It it like “paired” with an instance until this becomes unhealthy, then, it pairs to another healthy instance and send the request only to this instance.
- **ELB Cross Zone Balancing and Draining** is enable by default now but read carefully the question to see if the question reflects this.
- **The status for the ELB instances can be**:
  - **In service**
  - **Out of service**
  - **De-registration (draining, it is going out of service)**
- **ELB can be assigned a Security group** and can be configured to **forward the client/initiator's information including source IP address using X-Forwarded-For headers**
- When an ELB is deployed in a VPC, the ELB service itself launches ELB nodes (each ELB node requires an IP on the public subnet), depending on your traffic increase, the ELB service will continue to launch new ELB nodes eating up from your subnet available IP addresses, in the subnet where you defined the ELB in each AZ. **The more the traffic, the more IP addresses are taken from your subnet available IPs**
- **EC2 instances server by an internet-facing ELB can be in a public or private subnet and you don't need different subnet for the ELB** but you could have 2 private subnets for the web-servers, two public subnet for ELB and 2 private subnets for database in a Multi AZ architecture
- **Schema**
  - To configure an ELB to load balance to instances in a subnet in a specific AZ, the ELB must have one public subnet defined for the ELB in that AZ
  - Web instances that are being load balanced in an AZ can be on public or private subnets, as long as the ELB has a public subnet in that AZ defined as a load balancer subnet
  - To summarize the above, in order to have an ELB load balance between two AZs, it must have a public subnet in each AZ defined for it
  - Web instances can be in those two public subnets or in private subnets, it does not matter

AutoScaling Group (ASG)

- Auto Scaling provides the ability to ensure a correct number of EC2 instances are always running to handle the load of the application
- Auto Scaling helps to achieve better fault tolerance, better availability and cost management
- Auto Scaling also helps specify Scaling policies which can the be used to launch and terminate EC2 instances to handle any increase or decrease in demand on the application.
- Auto Scaling attempts to distribute instances evenly between the AZs that are enabled for the Auto Scaling group.
- Auto Scaling does this by attempting to launch new instances in the AZ with the fewest instances. If the attempt fails, Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds

Auto Scaling Configuration - Just review the schema if you don't want to read the next 6 pages :)



Launch Configuration

- Launch configuration is a **template** that an Auto Scaling group uses to launch EC2 instances.
- Launch configuration is similar to EC2 configuration and involves selection of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.
- Launch configuration can be associated multiple Auto Scaling groups
- **Launch configuration can't be modified** after creation and needs to be created new if any modification required
- Basic or Detailed monitoring for the instances in the Auto Scaling group can be enabled when a launch configuration is created.
- By default, **basic monitoring** is enabled when you create the launch configuration using the **AWS Management Console** and **detailed monitoring** is enabled when you create the launch configuration using the **Auto Scaling Group**
- Auto Scaling groups is the core of Auto Scaling and contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management.
- **Auto Scaling group requires**
    - Launch configuration to determine the EC2 template to use for launching the instance
    - Minimum & Maximum capacity to determine the number of instances when an auto-scaling policy is applied. The number of instances cannot grow beyond this boundaries
    - **Desired capacity**
        - To determine the number of instances the ASG must maintain at all times. If missing, it equals to the minimum size.
        - Desired capacity is different from minimum capacity.
        - An Auto Scaling group's desired capacity is the default number of instances that should be running. A group's minimum capacity is the fewest number of instances the group can have running
    - Availability Zones or Subnets in which the instances will be launched.
    - Metrics & Health Checks – metrics to determine when it should launch or terminate instances and health checks to determine if the instance is healthy or not
- Auto Scaling group starts by launching a desired capacity of instances and maintains this number by performing periodic health checks.
- If an instance becomes unhealthy, it terminates and launches a new instance
- Auto Scaling group can also use scaling policies to increase or decrease the number of instances automatically to meet changing demands
- An Auto Scaling group can contain EC2 instances in one or more AZs within the same region.

- Auto Scaling groups cannot span multiple regions.
- To merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple AZs, rezone one of the single-zone groups into a multi-zone group, and then delete the other groups. This process works for groups with or without a load balancer, as long as the new multi-zone group is in one of the same AZs as the original single-zone groups.
- Auto Scaling group can be associated with a single launch configuration.
- As the Launch Configuration can't be modified once created, only way to update the Launch Configuration for an Auto Scaling group is to create a new one and associate it with the Auto Scaling group.
- When the launch configuration for the Auto Scaling group is changed, any new instances launched use the new configuration parameters, but the existing instances are not affected.
- Auto Scaling group can be deleted from CLI, if it has no running instances else need to set the minimum and desired capacity to 0. This is handled automatically when deleting an ASG from AWS management console.

Auto Scaling Plans (Manual, Scheduled and Dynamic)

Maintain steady count of Instances

- Auto Scaling ensures a steady minimum (or desired if specified) count of Instances will always be running.
- If an instance is found unhealthy, Auto Scaling will terminate the Instance and launch a new one
- Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks
- Auto Scaling group can be associated with a load balancer enabled to use the Elastic Load Balancing health check, Auto Scaling determines the health status of the instances by checking the results of both EC2 instance status and Elastic Load Balancing instance health.
- Auto Scaling marks an instance unhealthy and launches a replacement if
    - the instance is in a state other than running,
    - the system status is impaired, or
    - Elastic Load Balancing reports the instance state as OutOfService.
- After an instance has been marked unhealthy as a result of an EC2 or ELB health check, it is almost immediately scheduled for replacement. It never automatically recovers its health.
- For an unhealthy instance, the instance's health check can be changed back to healthy manually but you will get an error if the instance is already terminating. Because the interval between marking an instance unhealthy and its actual termination is so small, attempting to set an instance's health status back to healthy is probably useful only for a suspended group
- When your instance is terminated, any associated Elastic IP addresses are disassociated and are not automatically associated with the new instance.
- Elastic IP addresses must be associated with the new instance manually.
- Similarly, when the instance is terminated, its attached EBS volumes are detached and must be attached to the new instance manually

Manual scaling (by adding/detaching EC2 or changing desired capacity manually)

- Manual scaling can be performed by
    - Changing the desired capacity limit of the Auto Scaling group
    - Attaching/Detaching instances to the Auto Scaling group
- Attaching/Detaching of an EC2 instance can be done only if
    - Instance is in the running state.
    - AMI used to launch the instance must still exist.
    - Instance is not a member of another Auto Scaling group.
    - Instance is in the same Availability Zone as the Auto Scaling group.
    - If the Auto Scaling group is associated with a load balancer, the instance and the load balancer must both be in the same VPC.
- Auto Scaling increases the desired capacity of the group by the number of instances being attached. But if the number of instances being attached plus the desired capacity exceeds the maximum size of the group, the request fails.
- When Detaching instances, you have the option of decrementing the desired capacity for the Auto Scaling group by the number of instances being detached. If you choose not to decrement the capacity, Auto Scaling launches new instances to replace the ones that you detached.
- If you detach an instance from an Auto Scaling group that is also registered with a load balancer, the instance is de-registered from the load balancer. If connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete.

Scheduled scaling (by scheduling the task by time)

- Scaling based on a schedule allows you to scale your application in response to predictable load changes for e.g. last day of the month, last day of an financial year
- Scheduled scaling requires configuration of Scheduled actions, which tells Auto Scaling to perform a scaling action at certain time in future, with the start time at which the scaling action should take effect, and the new minimum, maximum, and desired size the group should have
- Auto Scaling guarantees the order of execution for scheduled actions within the same group, but not for scheduled actions across groups
- Multiple Scheduled Actions can be specified but should have unique time value and they cannot have overlapping time scheduled which will lead to its rejection

Dynamic scaling (by alarms (CPU, etc)

- Allows you to scale automatically in response to the changing demand for e.g. scale out in case CPU utilization of the instance goes above 70% and scale in when the CPU utilization goes below 30%
- Auto Scaling group uses a combination of alarms & policies to determine when the conditions for scaling are met.

        - An alarm is an object that watches over a single metric over a specified time period. When the value of the metric breaches the defined threshold, for the number of specified time periods the alarm performs one or more actions (such as sending messages to Auto Scaling).
        - A policy is a set of instructions that tells Auto Scaling how to respond to alarm messages.
- **Dynamic scaling process works as below:**
    - Amazon CloudWatch monitors the specified metrics for all the instances in the Auto Scaling group
    - Changes are reflected in the metrics as the demand grows or shrinks (encoge)
    - When the change in the metrics breaches the threshold of the CloudWatch alarm, the CloudWatch alarm performs an action. Depending on the breach, the action is a message sent to either the scale-in policy or the scale-out policy
    - After the Auto Scaling policy receives the message, Auto Scaling performs the scaling activity for the Auto Scaling group.
    - This process continues until you delete either the scaling policies or the Auto Scaling group.

Multiple Policies (at least 2 policies per ASG)

- An Auto Scaling group can have more than one scaling policy attached to it any given time.
- Each Auto Scaling group would have at least two policies: one to scale the architecture out and another to scale the architecture in.
- If an Auto Scaling group has multiple policies, there is always a chance that both policies can instruct the Auto Scaling to Scale Out or Scale In at the same time.
- When this situations occur, Auto Scaling chooses the policy that has the greatest impact on the Auto Scaling group for e.g. if two policies are triggered at the same time and Policy 1 instructs to scale out the instance by 1 while Policy 2 instructs to scale out the instances by 2, Auto Scaling will use the Policy 2 and scale out the instances by 2 as it has a greater impact
-

Auto Scaling Cooldown (Ensure that ASG doesn’t launch or terminate additional instances before the previous scaling activity takes effect)

- Auto Scaling cool down period is a configurable setting for your Auto Scaling group that helps to ensure that Auto Scaling doesn’t launch or terminate additional instances before the previous scaling activity takes effect and allows the newly launched instances to start handling traffic and reduce load.
- When Auto Scaling group dynamically scales using a simple scaling policy and launches an instance, Auto Scaling suspends the scaling activities for the cool-down period (default 300 seconds) to complete before resuming scaling activities.

- **Example Use Case**

    1. You configure an scale out alarm to increase the capacity if the CPU utilization increases more then 80%
    2. An CPU spikes occurs and cause the alarm to be triggered, Auto Scaling launches a new instance
    3. However, it would take time for the newly launched instance to be configured, instantiated and started, lets say 5 mins
    4. Without a cooldown period, if an other CPU spikes occurs Auto Scaling would launch a new instance again and this would continue for 5 mins till the previously launched instance is up and running and started handling traffic
    5. With a cooldown period, Auto Scaling would suspend the activity for the specified time period enabling the newly launched instance to start handling traffic and reduce load
    6. After the cooldown period, Auto scaling resumes to act on the alarms
- When manually scaling the Auto Scaling group, the default is not to wait for the cool-down period, but you can override the default and honour the cooldown period.
- **Note that if an instance becomes unhealthy, Auto Scaling does not wait for the cooldown period to complete before replacing the unhealthy instance.**
- Cooldown periods are automatically applied to dynamic scaling activities for simple scaling policies and is not supported for step scaling policies.
- Termination Policy

Termination policy helps the Auto Scaling to decide which instances it should terminate first when Auto Scaling automatically scales in. Auto Scaling specifies a default termination policy and also allows you to create a customized one

Default Termination Policy (oldest launch configuration and unprotected are terminated first)

Default termination policy is designed to help ensure that the network architecture spans Availability Zones evenly and instances are selected for termination as follows :-

1. Selection of Availability Zone
    - Selects the AZ, in multiple AZs environment, with the most instances and at least one instance that is not protected from scale in.
    - Selects the AZ with instances that use the oldest launch configuration, if there are more than one AZ with same number of instances
2. Selection of an Instance in the Availability Zone
    - Terminates the unprotected instance using the oldest launch configuration, if one exists.
    - Terminates unprotected instances closest to the next billing hour, If multiple instances with oldest launch configuration. This helps in maximizing the use of the EC2 instances while minimizing the number of hours billed for EC2 usage.
    - Terminates instances at random, if more than one unprotected instance closest to the next billing hour
    -

Customized Termination Policy

1. Auto Scaling first assesses the Azs for any imbalance. If an AZ has more instances than the other AZs that are used by the group, then it applies the specified termination policy on the instances from the imbalanced AZ
2. If the Availability Zones used by the group are balanced, then Auto Scaling applies the termination policy that you specified.
3. Following Customized Termination policies are supported
    1. OldestInstance – terminates the oldest instance in the group and can be useful to upgrade to new instance types
    2. NewestInstance – terminates the newest instance in the group and can be useful when testing a new launch configuration
    3. OldestLaunchConfiguration – terminates instances that have the oldest launch configuration
    4. ClosestToNextInstanceHour – terminates instances that are closest to the next billing hour and helps to maximize the use of your instances and manage costs.
    5. Default – terminates as per the default termination policy

Instance Protection

- Instance protection controls whether Auto Scaling can terminate a particular instance or not.
- Instance protection can be enabled on a Auto Scaling group or an individual instance as well, at any time
- Instances launched within an Auto Scaling group with Instance protection enabled would inherit the property.
- Instance protection starts as soon as the instance is **InService** and if the Instance is detached, it loses its Instance protection
- If all instances in an ASG are protected from termination during scale in and a scale-in event occurs, it can’t terminate any instance and will decrement the desired capacity.
- Instance protection does not protect for the below cases
    - Manual termination through the EC2 console, the terminate-instances command, or the TerminateInstances API.
    - Termination if it fails health checks and must be replaced
    - Spot instances in an Auto Scaling group from interruption

Standby State (to either troubleshoot an instance or update an instance and return the instance back to service)

Auto Scaling allows you to put the InService instance in the Standby state during which the instance is still a part of the ASG but does not serve any requests. This can be used to either troubleshoot an instance or update an instance and return the instance back to service

- An instance can be put into Standby state and it will continue to remain in the Standby state unless exited
- Auto Scaling, by default, decrements the desired capacity for the group and prevents it from launching a new instance. If no decrement is selected, it would launch a new instance
- When the instance is in the standby state, instance can be updated or used for troubleshooting

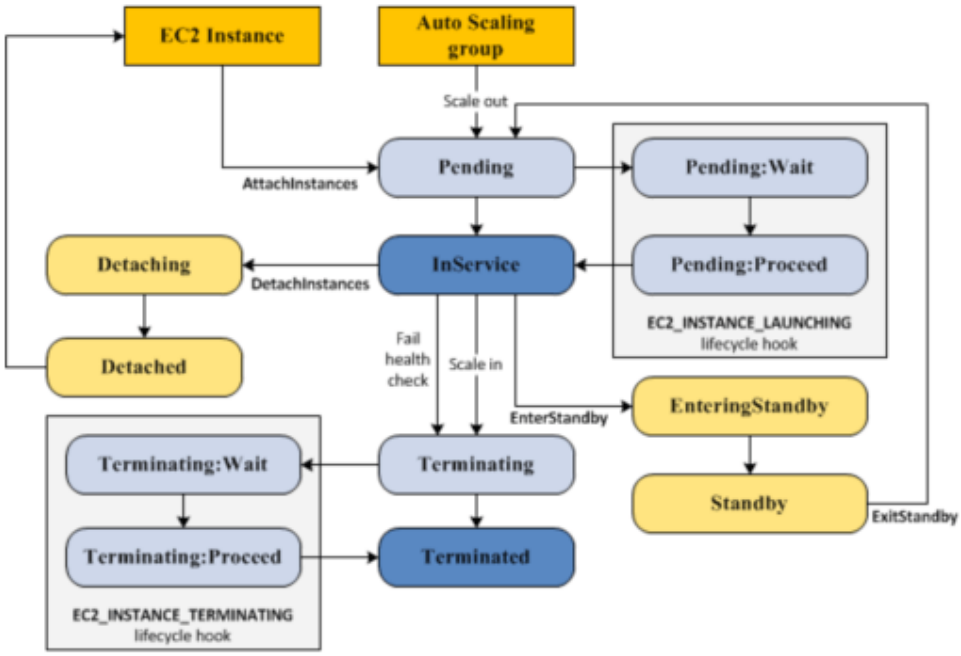
- If a load balancer is associated with Auto Scaling, the instance is automatically de-registered when the instance is in Standby state and registered again when the instance exits the Standby state

Suspension (to investigate a configuration problem or debug an issue with the application, without triggering the Auto Scaling process.)

- Auto Scaling processes can be suspended and then resumed. This can be very useful to investigate a configuration problem or debug an issue with the application, without triggering the Auto Scaling process.
- Auto Scaling also performs Administrative Suspension where it would suspend processes for ASGs, if the Auto Scaling groups that have been trying to launch instances for over 24 hours but have not succeeded in launching any instances.
- Auto Scaling processes include
    - Launch – Adds a new EC2 instance to the group, increasing its capacity.
    - Terminate – Removes an EC2 instance from the group, decreasing its capacity.
    - HealthCheck -Checks the health of the instances.
    - ReplaceUnhealthy – Terminates instances that are marked as unhealthy and subsequently creates new instances to replace them.
    - AlarmNotification – Accepts notifications from CloudWatch alarms that are associated with the group. If suspended, Auto Scaling does not automatically execute policies that would be triggered by an alarm
    - ScheduledActions – Performs scheduled actions that you create.
    - AddToLoadBalancer – Adds instances to the load balancer when they are launched.
    - AZRebalance – Balances the number of EC2 instances in the group across the Availability Zones in the region.
        - If an AZ either is removed from the ASG or becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected AZ before terminating the unhealthy or unavailable instances
        - When the unhealthy AZ returns to a healthy state, Auto Scaling automatically redistributes the instances evenly across the Availability Zones for the group.
        - Note that if you suspend AZRebalance and a scale out or scale in event occurs, Auto Scaling still tries to balance the Availability Zones for e.g. during scale out, it launches the instance in the Availability Zone with the fewest instances.
        - If you suspend Launch, AZRebalance neither launches new instances nor terminates existing instances. This is because AZRebalance terminates instances only after launching the replacement instances.
        - If you suspend Terminate, the ASG can grow up to 10% larger than its maximum size, because Auto Scaling allows this temporarily during rebalancing activities. If it cannot terminate instances, your ASG could remain above its maximum size until the Terminate process is resumed

Auto Scaling Lifecycle

- Instances launched through Auto Scaling group have a a different lifecycle then that of other EC2 instances
- Auto Scaling lifecycle starts when the Auto Scaling group launches an instance and puts it into service.
- Auto Scaling lifecycle ends when the instance is terminated either by the user , or the Auto Scaling group takes it out of service and terminates it
- AWS charges for the instances as soon as they are launched, including the time it is not in InService



- Auto Scaling Lifecycle Hooks
  - Auto Scaling Lifecycle hooksenable you to perform custom actions by pausing instances as an Auto Scaling group launches or terminates them
  - Each Auto Scaling group can have multiple lifecycle hooks. However, there is a limit on the number of hooks per Auto Scaling group
  - Auto Scaling scale out event flow
    - Instances start in the Pending state
    - If an autoscaling:EC2_INSTANCE_LAUNCHING lifecycle hook is added, the state is moved to Pending:Wait
    - After the lifecycle action is completed, instances enter to Pending:Proceed
    - When the instances are fully configured, they are attached to the Auto Scaling group and moved to the InService state

  - **Auto Scaling scale in event flow**

    - Instances are detached from the Auto Scaling group and enter the Terminating state.
    - If an autoscaling:EC2_INSTANCE_TERMINATING lifecycle hook is added, the state is moved to Terminating:Wait

        - After the lifecycle action is completed, the instances enter the Terminating:Proceed state.
        - When the instances are fully terminated, they enter the Terminated state.

- During the scale out and scale in events, instances are put into a wait state (Pending:Wait or Terminating:Wait) and is paused until either a continue action happens or the timeout period ends.
- By default, the instance remains in a wait state for one hour, which can be extended by restarting the timeout period by recoding a heartbeat. If the task finishes before the timeout period ends, the lifecycle action can be marked completed and it continues the launch or termination process.
- After the wait period the Auto Scaling group continues the launch or terminate process (Pending:Proceed or Terminating:Proceed)
    - **CloudWatch Events target to invoke a Lambda function when a lifecycle action occurs. Event contains information about the instance that is launching or terminating, and a token that can be used to control the lifecycle action.**

    - **Notification target (CloudWatch events, SNS, SQS) for the lifecycle hook which receives the message from EC2 Auto Scaling.The message contains information about the instance that is launching or terminating, and a token that you can use to control the life-cycle action.**

    - **Create a script that runs on the instance as the instance starts. The script can control the lifecycle action using the ID of the instance on which it runs. Custom action can be implemented using**

Auto Scaling Lifecycle Hooks Considerations

- Keeping Instances in a Wait State
    - Instances remain in a wait state for a finite period of time. Default being 1 hour (3600 seconds) with max being 48 hours or 100 times the heartbeat time-out, whichever is smaller.
    - Time can be adjusted using
        - complete-lifecycle-action (CompleteLifecycleAction) command to continue to the next state if finishes before the timeout period ends
        - put-lifecycle-hook command, the –heartbeat-timeout parameter to set the heartbeat timeout for the lifecycle hook during its creation
        - Restart the timeout period by recording a heartbeat, using the record-lifecycle-action-heartbeat (RecordLifecycleActionHeartbeat) command

- Cooldowns and Custom Actions
    - Cooldown period helps ensure that the Auto Scaling group does not launch or terminate more instances than needed

        - Cooldown period starts when the instance enters the InService state. Any suspended scaling actions resume after cooldown period expires
    - Health Check Grace Period
        - Health check grace period does not start until the lifecycle hook completes and the instance enters the InService state
    - Lifecycle Action Result
        - Result of lifecycle hook is either ABANDON or CONTINUE
        - If the instance is launching,
            - CONTINUE indicates a successful action, and the instance can be put into service.
            - ABANDON indicates the custom actions were unsuccessful, and that the instance can be terminated.
        - If the instance is terminating,
            - ABANDON and CONTINUE allow the instance to terminate.
            - However, ABANDON stops any remaining actions from other lifecycle hooks, while CONTINUE allows them to complete
    - Spot Instances
        - Lifecycle hooks can be used with Spot Instances. However, a lifecycle hook does not prevent an instance from terminating due to a change in the Spot Price, which can happen at any time

- ## Auto Scaling – Key notes

- Fault Tolerance is the ability of a system to remain in operation even if some of the components used to build the system fail. In AWS, this means that in the event of server fault or system failures, the number of running EC2 instances should not fall below the minimum number of instances required by the system for it to work properly. So if the the application requires a minimum of 4 instances, there should be at least 4 instances running in case there is an outage in one of the Availability Zones or if there are server issues.
- One of the differences between Fault Tolerance and High Availability is that, the former refers to the minimum number of running instances. For example, you have a system that requires a minimum of 4 running instances and currently has 6 running instances deployed in two Availability Zones. There was a component failure in one of the Availability Zones which knocks out 3 instances. In this case, the system can still be regarded as Highly Available since there are still instances running that can accommodate the requests. However, it is not Fault Tolerant since the required minimum of four instances have not been met.

AWS Relation Database Service

- **RDS features & benefits**
  - CPU, memory, storage, and IOPS can be scaled independently.
  - Manages backups, software patching, automatic failure detection, and recovery.
  - Automated backups can be performed as needed, or manual backups can be triggered as well. Backups can be used to restore a database, and the RDS restore process works reliably and efficiently.
  - Provides high availability with a primary instance and a synchronous secondary instance that can be failovered to seamlessly when a problem occurs.
  - Provides elasticity & scalability by enabling MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling.
  - **Supports MySQL Server, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server**, and the new, **MySQL-compatible Amazon Aurora DB engine**
  - In addition to the security in the database package, IAM users and permissions can help to control who has access to the RDS databases
  - Databases can be further protected by putting them in a VPC, using SSL for data in transit and encryption for data in rest
  - However, as it is a managed service, shell (root ssh) access to DB instances is not provided, and this restricts access to certain system procedures and tables that require advanced privileges.

RDS Components

- DB Instance
  - is a basic building block of RDS
  - is an isolated database environment in the cloud
  - each DB instance runs a DB engine. A**WS currently supports MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server & Aurora DB engines**
  - can be accessed from Amazon AWS command line tools, Amazon RDS APIs, or the AWS Management RDS Console.
  - computation and memory capacity of an DB instance is determined by its DB instance class, which can be selected as per the needs
  - for each DB instance, 5 GB to 6 TB of associated storage capacity can be selected
  - **storage comes in three types: Magnetic, General Purpose (SSD), and Provisioned IOPS (SSD), which differ in performance characteristics and price**
  - **each DB instance has a DB instance identifier, which is customer-supplied name and must be unique for that customer in an AWS region.** It uniquely identifies the DB instance when interacting with the Amazon RDS API and AWS CLI commands.
  - **Each DB instance can host multiple databases, or a single Oracle database with multiple schemas.**
  - It can be hosted in an AWS VPC environment for better control.

- Regions and Availability Zones
    - AWS resources are housed in highly available data center facilities in different areas of world, these data centers are called regions which further contain multiple distinct locations called Availability Zones
    - Each AZ is engineered to be isolated from failures in other AZs, and to provide inexpensive, low-latency network connectivity to other AZs in the same region
    - DB instances can be hosted in several AZs, an option called a Multi-AZ deployment.
        - Amazon automatically provisions and maintains a synchronous standby replica of the DB instance in a different AZ.
        - Primary DB instance is synchronously replicated across AZs to the standby replica
        - Provides data redundancy, fail-over support, eliminate I/O freezes, and minimize latency spikes during system backups.
- Security Groups
    - security group controls the access to a DB instance, by allowing access to the specified IP address ranges or EC2 instances
- DB Parameter Groups
    - A DB parameter group contains engine configuration values that can be applied to one or more DB instances of the same instance type
- DB Option Groups
    - Some DB engines offer tools that simplify managing the databases and making the best use of data.
    - Amazon RDS makes such tools available through option groups for e.g. Oracle Application Express (APEX), SQL Server Transparent Data Encryption, and MySQL memcached support.

RDS Interfaces

- RDS can be interacted with multiple interfaces
    - AWS RDS Management console
    - Command Line Interface
    - Programmatic Interfaces which include SDKs, libraries in different languages, and RDS API

RDS Pricing

- Instance class
    - Pricing is based on the class (e.g., micro, small, large, xlarge) of the DB instance consumed.
- Running time
    - Billed by the instance-hour, which is equivalent to a single instance running for an hour for e.g., a single instance running for two hours = two instances running for one hour, both consume 2 instance-hours.

        - if a DB instance runs for only part of an hour, full instance-hour is charged
- Storage
    - Storage capacity provisioned for the DB instance is billed per GB per month.
    - If the provisioned storage capacity is scaled within the month, the bill will be pro-rated.
- I/O requests per month
    - Total number of storage I/O requests made in a billing cycle.
- Backup storage
    - Automated backups & any active database snapshots consume storage
    - Increasing backup retention period or taking additional database snapshots increases the backup storage consumed by the database.
    - RDS provides backup storage up to 100% of the provisioned database storage at no additional charge for e.g., if you have 10 GB-months of provisioned database storage, RDS provides up to 10 GB-months of backup storage at no additional charge.
    - Most databases require less raw storage for a backup than for the primary dataset, so if multiple backups are not maintained, you will never pay for backup storage.
    - Backup storage is free only for active DB instances.
- Data transfer
    - Internet data transfer in and out of your DB instance.
- Reserved Instance
    - In addition to regular RDS pricing, reserved DB instances can be purchased

RDSMulti-AZ & Read Replica

- DB instances replicas can be created in two ways Multi-AZ & Read Replica
- Multi-AZ deployment
    - Multi-AZ deployment provides high availability and failover support
    - RDS automatically provisions and manages a synchronous standby replica in a different AZ (independent infrastructure in a physically separate location)
    - RDS automatically fails over to the standby so that database operations can resume quickly without administrative intervention in case of
        - Planned database maintenance
        - Software patching
        - Rebooting of the Primary instance
        - Primary DB instance connectivity or host failure, or an
        - Availability Zone failure
- Read Replica
    - RDS uses the **PostgreSQL, MySQL, and MariaDB DB engines**' built-in replication functionality to create a special type of DB instance called a Read Replica from a source DB instance.
    - Load on the source DB instance can be reduced **by routing read queries from applications to the Read Replica.**

        - Read Replicas allow elastic scaling beyond the capacity constraints of a single DB instance for read-heavy database workloads

Multi-AZ deployment

- Multi-AZ deployments provides high availability and automatic failover support for DB instances
- Multi-AZ helps improve the durability and availability of a critical system, enhancing availability during planned system maintenance, DB instance failure and Availability Zone disruption.
- Multi-AZ is a High Availability feature is not a scaling solution for read-only scenarios; standby replica can’t be used to serve read traffic. To service read-only traffic, use a Read Replica.
- Multi-AZ deployments for Oracle, PostgreSQL, MySQL, and MariaDB DB instances use Amazon technology, while SQL Server DB instances use SQL Server Mirroring.
- In a Multi-AZ deployment,
    - RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone.
    - Copies of data are stored in different Availability Zones for greater levels of data durability.
    - Primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide
        - data redundancy,
        - eliminate I/O freezes during snapshots and backups
        - and minimize latency spikes during system backups.
    - DB instances may have increased write and commit latency compared to a Single AZ deployment, due to the synchronous data replication
    - Transaction success is returned only if the commit is successful both on the primary and the standby DB
    - There might be a change in latency if the deployment fails over to the standby replica, although AWS is engineered with low-latency network connectivity between Availability Zones.
- **When using the BYOL licensing model, a license for both the primary instance and the standby replica is required**
- For production workloads, it is recommended to use Multi-AZ deployment with Provisioned IOPS and DB instance classes (m1.large and larger), optimized for Provisioned IOPS for fast, consistent performance.
- When Single-AZ deployment is modified to a Multi-AZ deployment (for engines other than SQL Server or Amazon Aurora)
    - RDS takes a snapshot of the primary DB instance from the deployment and restores the snapshot into another Availability Zone.
    - RDS then sets up synchronous replication between the primary DB instance and the new instance.
    - This avoids downtime when conversion from Single AZ to Multi-AZ

RDS Multi-AZ Failover Process

- In the event of a planned or unplanned outage of the DB instance,

        - RDS automatically switches to a standby replica in another AZ, if enabled for Multi-AZ.
        - Time it takes for the failover to complete depends on the database activity and other conditions at the time the primary DB instance became unavailable.
        - Fail-over times are typically 60-120 secs. However, large transactions or a lengthy recovery process can increase fail-over time.
        - Failover mechanism automatically changes the DNS record of the DB instance to point to the standby DB instance.
        - Multi-AZ switch is seamless to the applications as there is no change in the endpoint URLs but just needs to re-establish any existing connections to the DB instance.
    - **RDS handles failover automatically so that database operations can be resumed as quickly as possible without administrative intervention.**
    - Primary DB instance switches over automatically to the standby replica if any of the following conditions occur:
        - An Availability Zone outage
        - Primary DB instance fails
        - DB instance's server type is changed
        - Operating system of the DB instance is undergoing software patching
        - A manual fail over of the DB instance was initiated using Reboot with fail-over (also referred to as Forced Failover)
    - If the Multi-AZ DB instance has failed over, can be determined by
        - DB event subscriptions can be setup to notify you via email or SMS that a failover has been initiated.
        - DB events can be viewed via the Amazon RDS console or APIs.
        - Current state of your Multi-AZ deployment can be viewed via the RDS console and APIs.

Read Replica

- Amazon RDS uses the MySQL, MariaDB, and PostgreSQL DB engines' built-in replication functionality to create a Read Replica from a source DB instance.
- Updates made to the source DB instance are asynchronously copied to the Read Replica.
- **Load on the source DB instance can be reduced by routing read queries from the applications to the Read Replica.**
- Using Read Replicas allow DB to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- Read Replica operates as a DB instance that allows read-only connections; applications can connect to a Read Replica the same way they would to any DB instance.

Read Replica creation

- Up to five Read Replicas can be created from one source DB instance.
- Creation process

    - Automatic backups must be enabled on the source DB instance by setting the backup retention period to a value other than 0
    - Existing DB instance needs to be specified as the source.
    - RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot.
    - RDS then uses the asynchronous replication method for the DB engine to update the Read Replica for any changes to the source DB instance.
- RDS replicates all databases in the source DB instance.
- RDS sets up a secure communications channel between the source DB instance and the Read Replica, if that Read Replica is in a different AWS region from the DB instance.
- RDS establishes any AWS security configurations, such as adding security group entries, needed to enable the secure channel.
- During the Read Replica creation, a brief I/O suspension on the source DB instance can be experienced as the DB snapshot occurs.
- I/O suspension typically lasts about one minute and can be avoided if the source DB instance is a Multi-AZ deployment (in the case of Multi-AZ deployments, DB snapshots are taken from the standby).
- Read Replica creation time can be slow if any long-running transactions are being executed and should wait for completion
- For multiple Read Replicas created in parallel from the same source DB instance, only one snapshot is taken at the start of the first create action.
- A Read Replica can be promoted to a new independent source DB, in which case the replication link is broken between the Read Replica and the source DB. However, the replication continues for other replicas using the original source DB as the replication source

Read Replica Deletion & DB Failover

- Read Replicas must be explicitly deleted, using the same mechanisms for deleting a DB instance.
- If the source DB instance is deleted without deleting the replicas, each replica is promoted to a stand-alone, single-AZ DB instance.
- If the source instance of a Multi-AZ deployment fails over to the standby, any associated Read Replicas are switched to use the secondary as their replication source.

Read Replica Storage & Compute requirements

- A Read Replica, by default, is created with the same storage type as the source DB instance.
- For replication to operate effectively, each Read Replica should have the same amount of compute & storage resources as the source DB instance.
- Source DB instance, if scaled, Read Replicas should be scaled accordingly

Read Replica Features & Limitations

- RDS does not support circular replication.
- DB instance cannot be configured to serve as a replication source for an existing DB instance; a new Read Replica can be created only from an existing DB instance for e.g., if MyDBInstance replicates to ReadReplica1,

ReadReplica1 can't be configured to replicate back to MyDBInstance. From ReadReplica1, only a new Read Replica can be created, such as ReadRep2.

- Cross-Region Replication
  - MySQL, PostgresSQL or MariaDB Read Replica can be created in a different region than the source DB instance which helps to improve
    - disaster recovery capabilities (reduces RTO and RPO),
    - scale read operations into a region closer to end users,
    - migration from a data center in one region to another region
- Read Replica can be created from other Read replicas as well. However, the replica lag is higher for these instances and there cannot be more than four instances involved in a replication chain.

Feature/Behavior	PostgreSQL	MySQL and MariaDB
What is the replication method?	Physical replication.	Logical replication.
How are transaction logs purged?	PostgreSQL has a parameter, `wal_keep_segments`, that dictates how many Write Ahead Log (WAL) files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep.	Amazon RDS won't delete any binary logs that have not been applied.
Can a replica be made writable?	No. A PostgreSQL Read Replica is a physical copy and PostgreSQL doesn't allow for a Read Replica to be made writeable.	Yes. You can enable the MySQL or MariaDB Read Replica to be writable.
Can backups be performed on the replica?	Yes, you can create a snapshot of a PostgreSQL Read Replica, but you cannot enable automatic backups.	Yes. You can enable automatic backups on a MySQL or MariaDB Read Replica.
Can you use parallel replication?	No. PostgreSQL has a single process handling replication.	Yes. MySQL version 5.6 and all supported MariaDB versions allow for parallel replication threads.

Read Replica Use cases

- Read Replicas can be used in variety of use cases, including:
  - Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads, directing excess read traffic to Read Replica(s)
  - Serving read traffic while the source DB instance is unavailable for e.g. If the source DB instance cannot take I/O requests due to backups I/O suspension or scheduled maintenance, the read traffic can be directed to the Read Replica(s). However, the data might be stale.
  - Business reporting or data warehousing scenarios where business reporting queries can be executed against a Read Replica, rather than the primary, production DB instance.

AWS RDS Security

- AWS provides multiple features to provide RDS security
    - DB instance can be hosted in a VPC for the greatest possible network access control
    - IAM policies can be used to assign permissions that determine who is allowed to manage RDS resources
    - Security groups allow to control what IP addresses or EC2 instances can connect to the databases on a DB instance
    - Secure Socket Layer (SSL) connections with DB instances
    - RDS encryption to secure RDS instances and snapshots at rest.
    - Network encryption and transparent data encryption (TDE) with Oracle DB instances

RDS Authentication and Access Control

- IAM can be used to control which RDS operations each individual user has permission to call

Encrypting RDS Resources

- RDS encrypted instances use the industry standard AES-256 encryption algorithm to encrypt data on the server that hosts the RDS instance
- RDS then handles authentication of access and decryption of this data with a minimal impact on performance, and with no need to modify your database client applications
- **Data at Rest Encryption**
    - can be enabled on RDS instances to encrypt the underlying storage
    - encryption keys are managed by KMS
    - can be enabled only during instance creation
    - once enabled, the encryption keys cannot be changed
    - if the key is lost, the DB can only be restored from the backup
- Once encryption is enabled for an RDS instance,
    - logs are encrypted
    - snapshots are encrypted
    - automated backups are encrypted
    - read replicas are encrypted
- Cross region replicas and snapshots copy does not work since the key is only available in a single region
- RDS DB Snapshot considerations
    - DB snapshot encrypted using an KMS encryption key can be copied
    - Copying an encrypted DB snapshot, results in an encrypted copy of the DB snapshot
    - When copying, DB snapshot can either be encrypted with the same KMS encryption key as the original DB snapshot, or a different KMS encryption key to encrypt the copy of the DB snapshot.
    - An unencrypted DB snapshot can be copied to an encrypted snapshot, a quick way to add encryption to a previously unencrypted DB instance.

        - Encrypted snapshot can be restored only to an encrypted DB instance
        - If a KMS encryption key is specified when restoring from an unencrypted DB cluster snapshot, the restored DB cluster is encrypted using the specified KMS encryption key
        - Copying an encrypted snapshot shared from another AWS account, requires access to the KMS encryption key used to encrypt the DB snapshot.
        - Because KMS encryption keys are specific to the region that they are created in, encrypted snapshot cannot be copied to another region
- Transparent Data Encryption (TDE)
    - Automatically encrypts the data before it is written to the underlying storage device and decrypts when it is read from the storage device
    - is supported by Oracle and SQL Server
        - Oracle requires key storage outside of the KMS and integrates with CloudHSM for this
        - SQL Server requires a key but is managed by RDS
- SSL to Encrypt a Connection to a DB Instance
- Encrypt connections using SSL for data in transit between the applications and the DB instance
- **Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when RDS provisions the instance.**
- SSL certificates are signed by a certificate authority. SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks
- While SSL offers security benefits, be aware that SSL encryption is a compute-intensive operation and will increase the latency of the database connection.
- RDS Security Groups
- Security groups control the access that traffic has **in** and **out** of a DB instance
- VPC security groups act like a firewall controlling network access to your DB instance.
- VPC security groups can be configured and associated with the DB instance to allow access from an IP address range, port, or EC2 security group
- Database security groups default to a “deny all” access mode and customers must specifically authorize network ingress.

Master User Account Privileges

- When you create a new DB instance, the default master user that used gets certain privileges for that DB instance
- Subsequently, other users with permissions can be created

Event Notification

- Event notifications can be configured for important events that occur on the DB instance

- Notifications of a variety of important events that can occur on the RDS instance, such as whether the instance was shut down, a backup was started, a fail-over occurred, the security group was changed, or your storage space is low can be received
- AWS RDS Monitoring & Notification
  - RDS integrates with CloudWatch and provides metrics for monitoring
  - CloudWatch alarms can be created over a single metric that sends an SNS message when the alarm changes state
  - RDS also provides SNS notification whenever any RDS event occurs

CloudWatch RDS Monitoring

- RDS DB instance can be monitored using CloudWatch, which collects and processes raw data from RDS into readable, near real-time metrics.
- The statistics are recorded for a period of two weeks, so that you can access historical information and gain a better perspective on how the service is performing.
- By default, RDS metric data is automatically sent to Amazon CloudWatch in 1-minute periods
- CloudWatch RDS Metrics
  - **BinLogDiskUsage** – Amount of disk space occupied by binary logs on the master. Applies to MySQL read replicas.
  - **CPUUtilization** – Percentage of CPU utilization.
  - **DatabaseConnections** – Number of database connections in use.
  - **DiskQueueDepth** – The number of outstanding IOs (read/write requests) waiting to access the disk.
  - **FreeableMemory** – Amount of available random access memory.
  - **FreeStorageSpace** – Amount of available storage space.
  - ReplicaLag – Amount of time a Read Replica DB instance lags behind the source DB instance. Applies to MySQL, MariaDB, and PostgreSQL Read Replicas.
  - **SwapUsage** – Amount of swap space used on the DB instance.
  - **ReadIOPS** – Average number of disk I/O operations per second.
  - **WriteIOPS** – Average number of disk I/O operations per second.
  - **ReadLatency** – Average amount of time taken per disk I/O operation.
  - **WriteLatency** – Average amount of time taken per disk I/O operation.
  - **ReadThroughput** – Average number of bytes read from disk per second.
  - **WriteThroughput** – Average number of bytes written to disk per second.
  - **NetworkReceiveThroughput** – Incoming (Receive) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication.
  - **NetworkTransmitThroughput** – Outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication.

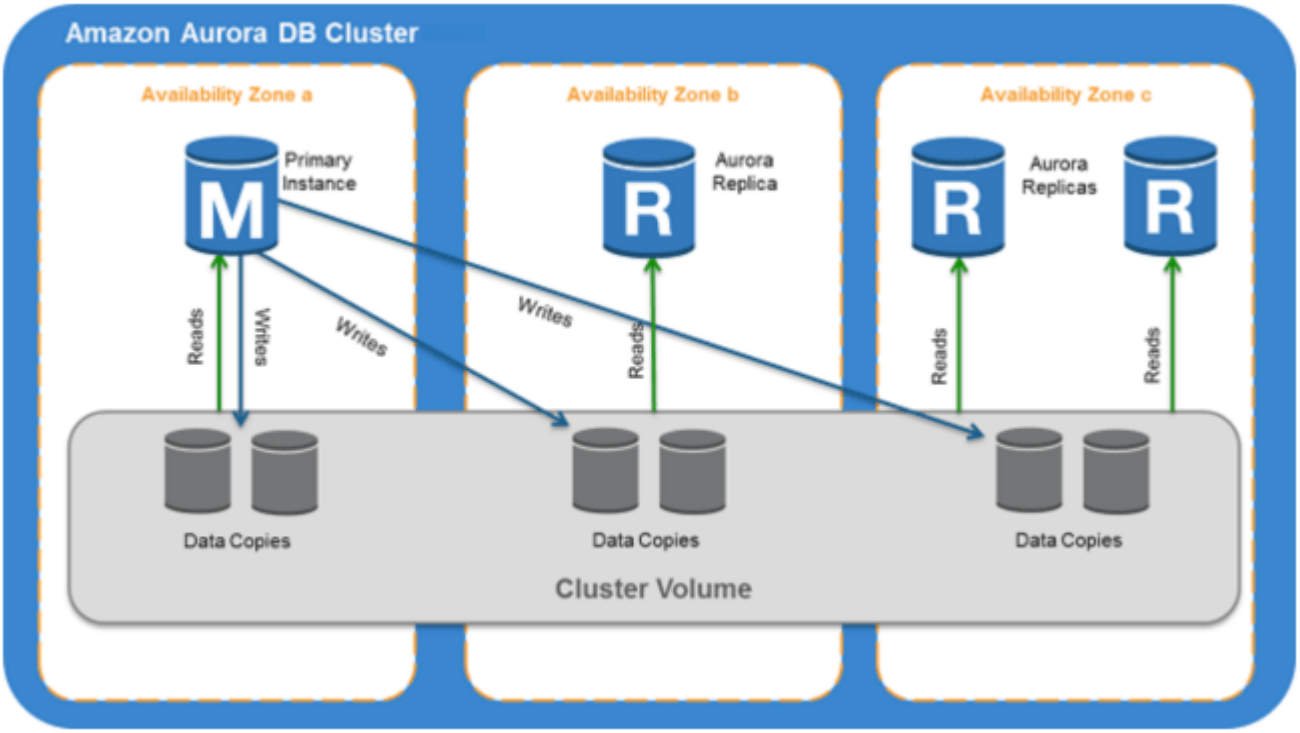
RDS Event Notification

- RDS uses the SNS to provide notification when an RDS event occurs
- RDS groups the events into categories, which can be subscribed so that a notification is sent when an event in that category occurs.
- Event category for a DB instance, DB cluster, DB snapshot, DB cluster snapshot, DB security group or for a DB parameter group can be subscribed
- Event notifications are sent to the email addresses provided during subscription creation
- **Subscription can be easily turn off notification without deleting a subscription by setting the Enabled radio button to No in the RDS console or by setting the Enabled parameter to false using the CLI or RDS API**

AWS Aurora

- AWS Aurora is a relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases.
- Amazon Aurora (Aurora) is a fully managed, MySQL- and PostgreSQL compatible, relational database engine i.e. applications developed with MySQL can switch to Aurora with little or no changes
- Aurora delivers up to 5x performance of MySQL without requiring any changes to most MySQL applications
- Aurora PostgreSQL delivers up to 3x performance of PostgreSQL.
- RDS manages the Aurora databases, handling time-consuming tasks such as provisioning, patching, backup, recovery, failure detection and repair.
- Based on the database usage, Aurora storage will automatically grow, from 10GB to 64TB in 10GB increments with no impact to database performance

NOTE – AWS Aurora is covered in the latest Solution Architect – Associate Feb 2018, so it is one of the important services to know :)



High Availability and Replication

- Aurora provides data durability and reliability by replicating the database volume six ways across three Availability Zones in a single region
    - Aurora automatically divides the database volume into 10GB segments spread across many disks.
    - Each 10GB chunk of your database volume is replicated six ways, across three Availability Zones
- RDS databases for e.g. MySQL, Oracle etc. have the data in a single AZ
- **Aurora is designed to transparently handle the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability.**
- Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and repaired automatically.
- Aurora Replicas share the same underlying volume as the primary instance. Updates made by the primary are visible to all Aurora Replicas
- As Aurora Replicas share the same data volume as the primary instance, there is virtually no replication lag
- **Any Aurora Replica can be promoted to become primary without any data loss and therefore can be used for enhancing fault tolerance in the event of a primary DB Instance failure.**
- To increase database availability, 1 to 15 replicas can be created in any of 3 AZs, and RDS will automatically include them in fail-over primary selection in the event of a database outage.

Security

- **Aurora uses SSL (AES-256) to secure the connection between the database instance and the application**
- Aurora allows database encryption using keys managed through AWS Key Management Service (KMS).
- Encryption and decryption are handled seamlessly.

- **With Aurora encryption, data stored at rest in the underlying storage is encrypted,** as are its automated backups, snapshots, and replicas in the same cluster.
- Encryption of existing unencrypted Aurora instance is not supported. Create a new encrypted Aurora instance and migrate the data

Backup and Restore

- **Automated backups are always enabled on Aurora DB Instances.**
- Backups do not impact database performance.
- Aurora also allows creation of manual snapshots
- Aurora automatically maintains 6 copies of your data across 3 AZs and will automatically attempt to recover your database in a healthy AZ with no data loss.
- If in any case the data is unavailable within Aurora storage,
    - DB Snapshot can be restored or
    - point-in-time restore operation can be performed to a new instance. Latest re-storable time for a point-in-time restore operation can be up to 5 minutes in the past.
- Restoring a snapshot creates a new Aurora DB instance
- Deleting Aurora database deletes all the automated backups (with an option to create a final snapshot), but would not remove the manual snapshots.
- Snapshots (including encrypted ones) can be shared with another AWS accounts

RDS Keypoints

Oracle SQL server import 20MB database
Oracle dataPump import complex DB
Control what IP or EC2 instance can connect to the DB with Security Group

- **AWS Databases:**
  - Supported RDBs:
    - Oracle
    - PostgreSQL
    - MS SQL Server
    - MySQL
    - MariaDB
    - AWS Aurora
      - Just to comment, don't see it:
        - **Operational Relational Databases**
          - Oracle
          - DB2
          - SQL Server
          - Mysql
          - Aurora
        - **Analytics Relational Databases**
          - Oracle
          - AWS Redshift **(Not RDS)**

1. **RDS Advantages**
   - Is an AWS fully managed Relational DB Engine service where AWS is responsible for:
     - Security and patching DB instances
     - Automated backup for your DB instance (default setting)
     - Software updates for the DB engineered
     - Easy scaling for storage and compute as required
     - If selected, Multi-AZ with Synchronous replication between the active and standby DB instances
     - Automatic fail over if Multi-AZ options was selected
     - Providing the ability to create DB read replicas for DB read scaling intensive read deployments
2. **RDS Disadvantages**
   - Inability to Scale Out (horizontally) to the needs of Web2.0 and Big Data APPS
   - Requires expensive hardware to scale up (vertically)
   - Requires more investment to span a distributed system
3. **Data Warehouse (Ex.: Alfresco APP)**
   - It is a collection of data integrated (collected) from multiple sources, which then undergoes complex long queries for analytical decisions and management reporting
4. **Maintenance**
   - Every DB instance has a weekly maintenance window. If you did not specify one at the time you create the DB instance AWS will chose one randomly for you (30 minutes long)

- Maintenance items require that Amazon RDS take your DB instance for a short time: For example:
  - Scale compute operations (few minutes)
  - OS or DB patching (it requires more time)

5. **Upgrades. AWS RDS Instances version upgrades**
   - Two types:
     - Major Version upgrades
       - AWS RDS does not do that automatically, you must perform all upgrades manually and an outage occurs while the upgrade takes placement
       - You cannot revert to the previous version
         - **For that, restore the DB snapshot that was taken before the upgrade to create a new DB instance**
     - Minor Version upgrades
6. **Licensing Models**
   - Two licensing Models:
     - Bring your Own License (BYOL)
     - Lincese included
7. **RDS Instance Storage**
   - **Amazon use EBS volumes**
     - **General Purpose RDS Storage:**
       - Use for DB workloads with moderate I/O requirements
     - **Provisioned IOPS RDS Storage:**
       - Use for High performance workloads
     - **Magnetic RDS Storage**
     - Use for Small DB workloads
8. **Understand the difference between**
   1. **OLTP** (Online Transaction Processing)
      - OLTP facilitates and manages transaction oriented applications (typically used in data entry and retrieval)
      - Database types: AWS RDS, AWS Aurora
      - An ATM machine transactions is an OLTP example
   2. **OLAP –** Online Analytic Processing
      - AWS Redshift
9. **Understand the difference between**
   - **RPO** (Recovery Point Objective) – the acceptable data loss.
   - **RTO** (Recovery Time Objective) – the time in future that your application can be live.

   1. You can specify a point-in-time restore any given second during the retention period
   2. When you initiate a point-in-time recovery, transaction DB logs are applied to the most appropriate daily backup to restore your DB to that point-in-time
   3. When you restore a DB instance, only the default DB parameters and Security groups are associated with the restored instance. Once the restore is complete, you need to associate/apply the customer DB parameters and security group settings

10. Manual DB Snapshots are not deleted automatically compared to Automated DB Snapshots!
11. **Restoring**

- You can not restore from a DB snapshot into an existing DB instance, rather it has to create a new DB instance with a new endpoint

12. Backups

- Automated backups are currently supported only for InnoDB storage engine for MySQL
- Although MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability
- To keep automatic backups enabled, retention period must be set to a non-zero value
- If you set retention period to Zero, automatic backups are disabled, thence, point-in time recovery which relies on automatic backups presence, will not function as well

13. Multi-AZ RDS

- It can be selected during RDS DB instance launch
- To create a fault tolerant and high available database architecture, implement Multi-AZ. Multi-AZ is a High Availability feature which helps protect against a failure to an AZ within a region, and when the primary requires to fail over to the Standby in another AZ (manual or automated fail over
    - But it can not protect against a region failure/outage since you cannot place the standby in a different region
        - Using read replicas come in handy **CrossRegion** in this case, since you can promote a read replica to a full RDS DB instance (standalone) or you can continue to read from it until the primary is brought up again
- Primary RDS DB instance servers all reads and writes
- RDS service creates a standby instance in a different AZ in the same region and configures **“SYNCHRONOUS”** replication between the primary and standby
- **Is not a scaling solution for read-only scenarios; Standby replica can’t be used to serve read traffic. To service read-only traffic, use a Read Replica.**
- Automated backups and Snapshots are taken from the standby RDS instance to avoid suspending I/O operations on the primary
- RDS endpoint does not change by selecting Multi-AZ option, however the primary and standby instances will have different IP addresses, given they are in different AZs. Use the DNS name in your application to connect to the database. If the database fails, AWS will update the records so it won’t impact your application. During the failover , the CNAME of the RDS DB instance is update to map to the standby IP address. The CNAME does not change, because the RDS endpoint does not change
    - Hence, recommended you don not use the IP address to point to your Multi-AZ RDS instances, rather use the endpoints.
        - This is very helpful in a fail-over scenario, since the Primary will fail to the standby, I.e IP address of RDS instance will change
            - Referencing by endpoint means, there is no change (end point wise) ehen fa fail-over happens, so no disruption or manual interventions to facilitate RDS DB instance access in a fail-over scenario

14. Use Read Replicas (scale read operations performance) in a heavy read traffic website. Load on the source DB instance can be reduced by routing read queries from the applications to the Read Replica. “Asynchronous” replication from Primary to read replica
    - You can promote a read replica into a standalone/single AZ database Instance
        - MySQL, MariaDB support read replicas of read replicas
    - Promotion process takes several minutes because the DB will reboot before the promoted replica becomes available as a standalone DB instance (allow for read/write, snapshots/backups, etc. ..)
    - The promoted replica into a standalone DB instance will retain:
        - Backup retention periodic
        - Backup window
        - DB parameter group of the formed replica source (primary instance)
15. You can test (safely) a DB instance against a new version before upgrading by:
    - Creating a DB snapshot (clone) of your existing DB instance,
    - Restoring from the DB snapshot to create a new DB instance
    - and then initiating a version upgrade for the new DB instance
    -

16. AWS RDS Events

- You need to subscribe to Amazon RDS events in order to be notified when/if changes occur with a DB instance, DB cluster, DB snapshot, DB cluster snapshot, DB parameter group or DB security group
- You can also Cloud Watch Alarms and events to monitor certain metrics and based on alarms or events take some actions (or notify using SNS)
- **CloudTrail** also logs all API calls made to the AWS RDS API

17. **RDS read replicas use InnoDB storage engine**
18. **AWS Auror**a is the database engine developed by AWS which is faster and cheaper than other AWS RDS.
    - Amazon Aurora is a relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. It delivers up to five times the throughput of standard MySQL and up to three times the throughput of standard PostgreSQL. Amazon Aurora is designed to be compatible with MySQL and with PostgreSQL, so that existing applications and tools can run without requiring modification. It is available through (RDS)
    - Amazon Aurora is designed to offer greater than 99.99% availability, increasing MySQL and PostgreSQL performance and availability by tightly integrating the database engine with an SSD-backed virtualized storage layer purpose-built for database workloads. Amazon Aurora's storage is fault-tolerant and self-healing, and disk failures are repaired in the background without loss of database availability. Amazon Aurora is designed to automatically detect database crashes and restart without the need for crash recovery or to rebuild the database cache. If the entire instance fails, Aurora delivers high performance and availability with up to 15 low-latency read replicas, point-in-time

recovery, continuous backup to Amazon S3, and replication across three Availability Zones.

- **Amazon Aurora scales storage automatically, growing storage and rebalancing I/Os to provide consistent performance without the need for over-provisioning. For example, you can start with a database of 10GB and have it automatically grow up to 64TB without requiring availability disruptions to resize or re-stripe data.**
- Based on the database usage, Aurora storage will automatically grow, from 10GB to 64TB in 10GB increments with no impact to database performance
- **Aurora provides data durability and reliability by replicating the database volume six ways across three Availability Zones in a single region**
  - Aurora automatically divides the database volume into 10GB segments spread across many disks.
  - Each 10GB chunk of your database volume is replicated six ways, across three Availability Zones
- **RDS databases for e.g. MySQL, Oracle etc. have the data in a single AZ while Aurora is multi AZ and it is designed to transparently handle the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability.**
- **Any Aurora Replica can be promoted to become primary without any data loss and therefore can be used for enhancing fault tolerance in the event of a primary DB Instance failure.**
- Restoring a snapshot creates a new Aurora DB instance
- Deleting Aurora database deletes all the automated backups (with an option to create a final snapshot), but would not remove the manual snapshots.
  - **Aurora Security**
    - **Aurora uses SSL (AES-256) to secure the connection between the database instance and the application**
    - Aurora allows database encryption using keys managed through AWS Key Management Service (KMS).
    - **With Aurora encryption, data stored at rest in the underlying storage is encrypted,** as are its automated backups, snapshots, and replicas in the same cluster.
    - Encryption of existing unencrypted Aurora instance is not supported. Create a new encrypted Aurora instance and migrate the data

19. **AWS Database Migration Service** helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from most widely used commercial and open-source databases.
    - The service supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle to Amazon Aurora or Microsoft SQL Server to MySQL. It also allows you to stream data to Amazon Redshift,

Amazon DynamoDB, and Amazon S3 from any of the supported sources, which are Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, SAP ASE, SQL Server, IBM DB2 LUW, and MongoDB, enabling consolidation and easy analysis of data in a petabyte-scale data warehouse. AWS Database Migration Service can also be used for continuous data replication with high-availability.

- When migrating databases to Aurora, Redshift or DynamoDB, you can use DMS free for six months.

AWS DynamoDB

- **Amazon DynamoDB is a fully managed NoSQL database service that:**
  - **makes it simple and cost-effective to store and retrieve any amount of data and serve any level of request traffic.**
  - **provides fast and predictable performance with seamless scalability**
- DynamoDB enables customers to offload the administrative burdens of operating and scaling distributed databases to AWS, without having to worry about hardware provisioning, set-up and configuration, replication, software patching, or cluster scaling.
- **DynamoDB tables do not have fixed schemas**, and table consists of items and each item may have a different number of attributes.
- **DynamoDB synchronously replicates data across three facilities** in an AWS Region, giving high availability and data durability.
- DynamoDB supports fast in-place updates. A numeric attribute can be incremented or decremented in a row using a single API call
- DynamoDB uses proven cryptographic methods to securely authenticate users and prevent unauthorized data access
- Durability, performance, reliability, and security are built in, w**ith SSD (solid state drive) storage and automatic 3-way replication.**
- DynamoDB supports two different kinds of primary keys:
  - Partition Key (previously called the Hash key)
    - A simple primary key, composed of one attribute
    - DynamoDB uses the partition key's value as input to an internal hash function; the output from the hash function determine the partition where the item will be stored.
    - No two items in a table can have the same partition key value.
  - Partition Key and Sort Key (previously called the Hash and Range key)
    - **A composite primary key composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key.**
    - DynamoDB uses the partition key value as input to an internal hash function; the output from the hash function determines the partition where the item will be stored.
    - All items with the same partition key are stored together, in sorted order by sort key value.

        - It is possible for two items to have the same partition key value, but those two items must have different sort key values.
- DynamoDB Secondary indexes
    - add flexibility to the queries, without impacting performance.
    - are automatically maintained as sparse objects, items will only appear in an index if they exist in the table on which the index is defined making queries against an index very efficient
- DynamoDB throughput and single-digit millisecond latency makes it a great fit for gaming, ad tech, mobile, and many other applications
- **ElastiCache can be used in front of DynamoDB in order to offload high amount of reads for non frequently changed data**
- DynamoDB Performance
- Automatically scales horizontally
- **runs exclusively on Solid State Drives (SSDs).**
    - SSDs help achieve the design goals of predictable low-latency response times for storing and accessing data at any scale.
    - SSDs High I/O performance enables it to serve high-scale request workloads cost efficiently, and to pass this efficiency along in low request pricing
- allows provisioned table reads and writes
    - Scale up throughput when needed
    - Scale down throughput four times per UTC calendar day
- automatically partitions, reallocates and re-partitions the data and provisions additional server capacity as the
    - table size grows or
    - provisioned throughput is increased
- Global Secondary indexes (GSI)
    - can be created upfront or added later

DynamoDB Consistency

- Each DynamoDB table is automatically stored in the three geographically distributed locations for durability
- Read consistency represents the manner and timing in which the successful write or update of a data item is reflected in a subsequent read operation of that same item
- DynamoDB allows user to specify whether the read should be eventually consistent or strongly consistent at the time of the request
    - Eventually Consistent Reads (Default)
        - Eventual consistency option maximizes the read throughput.
        - Consistency across all copies is usually reached within a second
        - However, an eventually consistent read might not reflect the results of a recently completed write.
        - Repeating a read after a short time should return the updated data.
    - Strongly Consistent Reads
        - Strongly consistent read returns a result that reflects all writes that received a successful response prior to the read

- Query, **GetItem**, and **BatchGetItem** operations perform eventually consistent reads by default
    - Query and GetItem operations can be forced to be strongly consistent
    - Query operations cannot perform strongly consistent reads on Global Secondary Indexes
    - BatchGetItem operations can be forced to be strongly consistent on a per-table basis

DynamoDB Security

- Fine Grained Access Control (FGAC) gives a high degree of control over data in the table
- FGAC helps control *who (*caller) can access *which* items or attributes of the table and perform *what* actions (read/write capability).
- FGAC is integrated with IAM, which manages the security credentials and the associated permissions.
- DynamoDB Encryption
- **Data in Transit Encryption**
    - can be done by encrypting sensitive data on the client side or using encrypted connections (TLS)
- **DynamoDB supports Encryption at rest**
    - Encryption at rest enables encryption for the data persisted (data at rest) in the DynamoDB tables.
    - Encryption at rest includes the base tables, secondary indexes
    - Encryption at rest automatically integrates with AWS KMS for managing the keys used for encrypting the tables.
    - Encryption at rest can be enabled only for a new table and not for an existing table
    - **Encryption once enabled for a table, cannot be disabled**
    - DynamoDB Streams do not support encryption
    - On-Demand Backups of encrypted DynamoDB tables are encrypted using S3’s Server-Side Encryption
    - Encryption at rest encrypts your data using 256-bit AES encryption.

DynamoDB Costs

- Index Storage
    - DynamoDB is an indexed data store
        - Billable Data = Raw byte data size + 100 byte per-item storage indexing overhead
- Provisioned throughput
    - Pay flat, hourly rate based on the capacity reserved as the throughput provisioned for the table
    - one Write Capacity Unit provides one write per second for items < 1KB in size.
    - one Read Capacity Unit provides one strongly consistent read (or two eventually consistent reads) per second for items < 4KB in size.
    - Provisioned throughput charges for every 10 units of Write Capacity and every 50 units of Read Capacity.

- **Reserved capacity**
    - Significant savings over the normal price
    - Pay a one-time upfront fee
    -

DynamoDB Cross-region Replication

- DynamoDB cross-region replication allows identical copies (called replicas) of a DynamoDB table (called master table) to be maintained in one or more AWS regions
- Writes to the table will be automatically propagated to all replicas
- Cross-region replication currently supports single master mode. A single master has one master table and one or more replica tables
- Read replicas are updated asynchronously as DynamoDB acknowledges a write operation as successful once it has been accepted by the master table. The write will then be propagated to each replica with a slight delay .
- **Cross-region replication can be helpful in scenarios:**
    - Efficient disaster recovery, in case a data center failure occurs.
    - Faster reads, for customers in multiple regions by delivering data faster by reading a DynamoDB table from the closest AWS data center.
    - Easier traffic management, to distribute the read workload across tables and thereby consume less read capacity in the master table.
    - Easy regional migration, by promoting a read replica to master
    - Live data migration, to replicate data and when the tables are in sync, switch the application to write to the destination region
- Cross-region replication costing depends on
    - Provisioned throughput (Writes and Reads)
    - Storage for the replica tables.
    - Data Transfer across regions
    - Reading data from DynamoDB Streams to keep the tables in sync.
    - Cost of EC2 instances provisioned, depending upon the instance types and region, to host the replication process.

- NOTE : Cross Region replication on DynamoDB was performed defining AWS Data Pipeline job which used EMR internally to transfer data before the DynamoDB streams and out of box cross region replication support

DynamoDB Global Tables

- DynamoDB Global Tables is a new multi-master, cross-region replication capability of DynamoDB to support data access locality and regional fault tolerance for database workloads.
- Applications can now perform reads and writes to DynamoDB in AWS regions around the world, with changes in any region propagated to every region where a table is replicated.
- Global Tables help in building applications to advantage of data locality to reduce overall latency.
- Global Tables ensures eventual consistency
- Global Tables replicates data among regions within a single AWS account, and currently does not support cross account access

DynamoDB Streams

- DynamoDB Streams provides a time-ordered sequence of item-level changes made to data in a table in the last 24 hours, after which they are erased
- DynamoDB Streams maintains ordered sequence of the events per item however across item are not maintained
- Example
    - For e.g., suppose that you have a DynamoDB table tracking high scores for a game and that each item in the table represents an individual player. If you make the following three updates in this order:
        - Update 1: Change Player 1’s high score to 100 points
        - Update 2: Change Player 2’s high score to 50 points
        - Update 3: Change Player 1’s high score to 125 points
    - DynamoDB Streams will maintain the order for Player 1 score events. However, it would not maintain the order across the players. So Player 2 score event is not guaranteed between the 2 Player 1 events
- DynamoDB streams can be used for multi-region replication to keep other data stores up-to-date with the latest changes to DynamoDB or to take actions based on the changes made to the table
- DynamoDB Streams APIs helps developers consume updates and receive the item-level data before and after items are changed
- DynamoDB Streams allows read at up to twice the rate of the provisioned write capacity of the DynamoDB table
- DynamoDB Streams have to be enabled on a per-table basis
- DynamoDB Streams is designed so that every update made to the table will be represented exactly once in the stream. No Duplicates

DynamoDB Triggers

- DynamoDB Triggers (just like database triggers) is a feature which allows execution of custom actions based on item-level updates on a table
- **DynamoDB triggers can be used in scenarios like sending notifications, updating an aggregate table, and connecting DynamoDB tables to other data sources**
- DynamoDB Trigger flow
    - **Custom logic for a DynamoDB trigger is stored in an AWS Lambda function as code.**
    - A trigger for a given table can be created by associating an AWS Lambda function to the stream (via DynamoDB Streams) on a table.
    - When the table is updated, the updates are published to DynamoDB Streams.
    - In turn, AWS Lambda reads the updates from the associated stream and executes the code in the function.

DynamoDB Accelerator DAX

- **DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement – from milliseconds to microseconds – even at millions of requests per second.**
- DAX does all the heavy lifting required to add in-memory acceleration to the tables, without requiring developers to manage cache invalidation, data population, or cluster management.
- DAX is fault-tolerant and scalable.
- DAX cluster has a primary node and zero or more read-replica nodes. Upon a failure for a primary node, DAX will automatically fail over and select a new primary. For scaling, add or remove read replicas

AWS DynamoDB Secondary Indexes

- **DynamoDB provides fast access to items in a table by specifying primary key values**
    - DynamoDB Secondary indexes on a table allow efficient access to data with attributes other than the primary key
    - **DynamoDB Secondary indexes:**
        - is a data structure that contains a subset of attributes from a table
        - is associated with exactly one table, from which it obtains its data
        - requires an alternate key for the index partition key and sort key
        - additionally can define projected attributes which are copied from the base table into the index along with the primary key attributes
        - is automatically maintained by DynamoDB
        - any addition, modification, or deletion of items in the base table, any indexes on that table are also updated to reflect these changes.
        - helps reduce the size of the data as compared to the main table, depending upon the project attributes and hence helps improve provisioned throughput performance

        - are automatically maintained as sparse objects. Items will only appear in an index if they exist in the table on which the index is defined, making queries an index very efficient
- **DynamoDB Secondary indexes supports two types**
    - **Global secondary index –** an index with a partition key and a sort key that can be different from those on the base table
    - **Local secondary index** – an index that has the same partition key as the base table, but a different sort key

Global Secondary Indexes (GSI)

- DynamoDB creates and maintains indexes for the primary key attributes for efficient access of data in the table, which allows applications to quickly retrieve data by specifying primary key values.
- **Global Secondary Indexes (GSI) are indexes that contain partition or composite partition-and-sort keys that can be different from the keys in the table on which the index is based.**
- **Global secondary index is considered "global" because queries on the index can span all items in a table, across all partitions.**
- Multiplesecondary indexes can be created on a table, and queries issued against these indexes.
- Applications benefit from having one or more secondary keys available to allow efficient access to data with attributes other than the primary key.
- GSIs support non-unique attributes, which increases query flexibility by enabling queries against any non-key attribute in the table
- GSIs support eventual consistency. DynamoDB automatically handles item additions, updates and deletes in a GSI when corresponding changes are made to the table asynchronously
- Data in a secondary index consists of GSI alternate key, primary key and attributes that are projected, or copied, from the table into the index.
- Attributes that are part of an item in a table, but not part of the GSI key, primary key of the table, or projected attributes are not returned on querying the GSI index
- GSIs manage throughput independently of the table they are based on and the provisioned throughput for the table and each associated GSI needs to be specified at creation time
    - Read provisioned throughput
        - provides one Read Capacity Unit with two eventually consistent reads per second for items < 4KB in size.
        - Provides one Write Capacity Uni with one write per second for items < 1KB in size.
    - Write provisioned throughput
        - consumes 1 write capacity unit if,
            - new item is inserted into table
            - existing item is deleted from table
            - existing items is updated for project attributes
        - consumes 2 write capacity units if
            - existing item is updated for key attributes, which results in deletion and addition of the new item into the index

Local Secondary Indexes

- Local secondary index are indexes that has the same partition key as the table, but a different sort key.
- Local secondary index is “local” cause every partition of a local secondary index is scoped to a table partition that has the same partition key.
- LSI allows search using a secondary index in place of the sort key, thus expanding the number of attributes that can be used for queries which can be conducted efficiently
- LSI are updated automatically when the primary index is updated and reads support both strong and eventually consistent options
- LSIs can only be queried via the Query API
- LSIs cannot be added to existing tables at this time
- LSIs cannot be modified once it is created at this time
- LSI cannot be removed from a table once they are created at this time
- LSI consumes provisioned throughput capacity as part of the table with which it is associated
    - Read Provisioned throughput
        - if data read is index and projected attributes
            - provides one Read Capacity Unit with one strongly consistent read (or two eventually consistent reads) per second for items < 4KB
            - data size includes the index and projected attributes only
        - if data read is index and a non projected attribute
            - consumes double the read capacity, with one to read from the index and one to read from the table with the entire data and not just the non projected attribute
    - Write provisioned throughput
        - consumes 1 write capacity unit if,
            - new item is inserted into table
            - existing item is deleted from table
            - existing items is updated for project attributes
        - consumes 2 write capacity units if
            - existing item is updated for key attributes, which results in deletion and addition of the new item into the index

	Global Secondary Index	Local Secondary Index
Search	Searches across entire table and all partitions	Search limited to a Single partition
Creation	Can be created at as well as after table creation	Can be created only at table creation
Deletion	Can be deleted at any time	Cannot be deleted independently
Read Consistency	Supports Eventual Consistency	Supports either Strong or Eventual Consistency
Size	No Restriction	Limited to 10GB per Partition
Throughput	Needs separate provisioned throughput	Consumes Tables provisioned throughput
Projected Attributes	Returns only Projects attributes configured during creation	Can request attributes that are not projected into the index
Key Schema	Primary key can be either simple (partition key) or composite (partition key and sort key)	Primary key must be composite (partition key and sort key)

DynamoDB Best Practices

- Keep item size small
- Store meta-data in DynamoDB and large BLOBs in Amazon S3
- Use table per day, week, month etc for storing time series data
- Use conditional or Optimistic Concurrency Control (OCC) updates
  - Optimistic Concurrency Control is like Optimistic locking in the RDMS
  - OCC is generally used in environments with low data contention, conflicts are rare and transactions can be completed without the expense of managing locks and transactions
  - OCC assumes that multiple transactions can frequently be completed without interfering with each other.
  - Transactions are executed using data resources without acquiring locks on those resources and waiting for other transaction locks to be cleared
  - Before a transaction is committed, it is verified if the data was modified by any other transaction. If so, it would be roll-backed and needs to be restarted with the updated data
  - OCC leads to higher throughput as compared to other concurrency control methods like pessimistic locking, as locking can drastically limit effective concurrency even when deadlocks are avoided
  - Avoid hot keys and hot partitions

DynamoDB - Keypoints

1. **DynamoDB is a fully managed AWS NoSQL database** and supports both document and key-value data models
    - Is extremely fast and delivers predictable performance with seamless scalability
    - Use for applications that need cosnsistent, single-digit millisecond latency at any scale
    - It is great for:
        - Mobile APPS
        - Web APPS
        - Gaming APPS
        - Ad-tech APPS
        - Internet of things (IoT)
    - Allows latency read/write access to items ranging from 1 byte to 400 bytes
    - Can be used to store pointers to S3 stored objects, or items of sizes larger than 400KBs too if needed
    - DynamoDB stores data indexed by a primary key
        - You specify the primary key when you create the table
    - Each item in the table has a unique identifier or primary key, that distinguishes the item from all of the others in the table
    - The primary key is the only required attribute for items in a table
    - Supports Get/Put operations using a user defined Primary Key
2. **Local secondary index** are indexes that has the same partition key as the table, but a different sort key.
    - Local secondary index is “local” cause every partition of a local secondary index is scoped to a table partition that has the same partition key.
    - DynamoDB supports two types of secondary indexes:
        - Global secondary index – an index with a partition key and a sort key that can be different from those on the base table
        - Local secondary index – an index that has the same partition key as the base table, but a different sort key
    - DynamoDB provides fast access to items in a table by specifying primary key values
    - DynamoDB Secondary indexes on a table allow efficient access to data with attributes other than the primary key
3. **FGAC:** end user or APP wants to read or modify the table directly
4. **DynamoDB tables are schema-less**:
    - Which means that neither the attributes nor their data types need to be defined beforehand
    - Each item can have its own distinct attributes
    - Like all other Dbs, DynamoDB stores data in tables
    -
5. **DynamoDB tables**
    - Like all other Dbs, DynamoDB stores data in tables
    - A table is a collection of data items
    - An item, is a group of attributes that is uniquely identifiable among all of the other items. Ex.: Artist (attribute): SongTitle (attribute)

- An attribute consists of the attribute name and a value or a set of values
- An Item consists of a primary or composite key and a flexible number of attributes
- Items in DynamoDB are similar in to rows
- There is not limit to the number of items you can store in a table
- Two items in a table can have the same primary key

- **Data model of dynamoDB**:
  - Table
    - Items
      - Attribute

6. **DynamoDB does not support**:
   - Complex relations DB querying or joins
   - Does not support complex transactions
7. **DynamoDB Performance**
   - Automatically replicates data across three facilities (Data Centers not AZs) in an AWS region for HA and data durability
     - It also partitions your DB over sufficient number of servers according to your read/write capacity
     - Performs automatic fail-over in case of any failure
   - Runs exclusively on SSD volumes which provides:
     - Low Latency
     - Predictable performance
     - High I/Os
8. **Scalability**
   - It provides for a push button scaling on AWS where you can increase the read/write throughput and AWS will go ahead and scale it for you (up or down) without downtime or performance degradations
   - You can scale the provisioned capacity of your DynamoDB table any time you want
   - You scale down your provisioned capacity only 4 times during a calendar day
   - Up to 10.000 write/read capacity units per second per table
     - If you need higher than this, contact AWS
9. **DynamoDB Throttling**
   - If you read or write request exceed the throughput settings for a table, DynamoDB can throttle that request
   - DynamoDB can also throttle read requests exceeds for an indexing
     - Throttling prevents your application from consuming too many capacity units
     - When a reques is throttled, it fails with an HTTP 400 code (bad request) and ProvisionedThroughputExceededExecption
   - The AWS SDKs have built-in support for retrying throttled requests
10. **Read Consistency**
    - **Eventually Consistent Reads (default)**
      - When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation
      - Best read throughput-intensive
      - Consistency across all copies is reached in 1 second
    - **Strongly Consistent reads:**

        - A read returns a result that reflects all writes that received a successful response prior to the read **(HTTP 200 = OK)**

11. **DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement** – from milliseconds to microseconds – even at millions of requests per second.DAX does all the heavy lifting required to add in-memory acceleration to the tables, without requiring developers to manage cache invalidation, data population, or cluster management.
12. ElastiCache can be used in front of DynamoDB in order to offload high amount of reads for non frequently changed data
13. Increase the write efficiency of an Amazon DynamoDB by randomizing the primary key value.
14. **DynamoDB Limits**
    - 256 tables per account per region
    - No limit on the size of any table
15. **DynamoDB – Integration with RedShift**
    - Loading data from DynamoDB into Amazon RedShift:
        - Amazon Redshift complements Amazon DynamoDB with advanced business intelligence capabilities and a powerful SQL-Based interfaces
        - When you copy data from a DynamoDB table into RedShift, you can perform complex data analysis queries on that data, including joins with other tables in your Redshift clustersIn terms of provisioned throughput, a copy operation from a DynamoDB table counts against that table's read capacity
        - After data is copied, your SQL queries in Redshift do not affect DynamoDB in any way
        - Before you can load data from a DynamoDB table, you must first create an Amazon RedShift table to serve as the destination for the data
        - Keep in mind that you are copying data from a NoSQL environment into a SQL environment, and that there are certain rules in one environment that do not apply in the other
16. **Non-Relational Databases**
    - Store data without a structured mechanism to link data from different tables to one another
    - **Are high performance databases that are non-schema** based unlike relational Dbs
    - Use non-structured or semi-structured data (JSON,XML)
    - Storage and retrieval of data is modelled without/away from tabular relations as in SQL Dbs
        - Use a variety of data models, including docuOpen Source NoSQL
        - Fastest NoSQLment, graph, key-value and columnar
    - They scale out (horizontally) using distributed clusters to increase throughput without increasing latency (This meets today's needs in social media, analytics, big data and IoT
    - Bes suited for On Line Analytical Processing (OLAP). Examples:
        - Business reporting for sales
        - Management reporting
        - Business process management

        - Financial reporting
        - Analytics
    - Examples of No-SQL Dbs:
        - **Analytics**
            - AWS Elastic Map Reduce (EMR) → Analytics (Hadoop Storage)
        - **Operational**
            - AWS DynamoDB
            - Casssandra
        - **Redis**
            - Open Source NoSQL
            - Fastest NoSQL

17. **Non-Relational Databases – Types**
    - **Columnar** (RedShift)
        - Optimizing for reading and writing columns not rows
        - Reduce the amount of data to be loaded from the disk
        - Scale out using distributed clusters (low cost hardware)
    - **Document Databases**
        - DynamoDB
    - **Graph Databases**
    - **In-memory key-value Databases** (Amazon Elastic Cache: Memcached | Redis)
        - An in-memory key-value store is a NoSQL DB optimized for read heavy application and intensive workloads. Examples:
            - Social networking, gamin, media sharing and Q&A portals
        - They improve application performance by storing critical pieces of data in memory for low-latency access

AWS Route 53

- **Amazon Route 53 provides three main functions:**
  - **Domain registration**
    - allows you to register domain names or transfer existing domains to Route53 Management
  - **DNS sevice (Domain Name System)**
    - translates friendly domains names like www.example.com into IP addresses like 192.0.2.1
    - responds to DNS queries using a global network of authoritative DNS servers, which reduces latency
    - can route Internet traffic to CloudFront, Elastic Beanstalk, ELB, or S3. There's no charge for DNS queries to these resources
  - **Health checking**
    - can monitor the health of resources such as web and email servers.
    - sends automated requests over the Internet to the application to verify that it's reachable, available, and functional
    - CloudWatch alarms can be configured for the health checks to send notification when a resource becomes unavailable.
    - can be configured to route Internet traffic away from resources that are unavailable

Route53 Supported DNS Record Types:

A	Maps a host to an IPv4 address. (decimal notation for e.g. 192.0.2.1)
AAAA	**Maps a host to an IPv6 address**. (colon-separated hexadecimal format)
CNAME	**Creates an alias to a domain name**. It is the same format as a domain name: • DNS protocol does not allow creation of a CNAME record for the top node of a DNS name-space, also known as the zone apex for e.g. the DNS name example.com registration, the zone apex is example.com, a CNAME record for example.com cannot be created, but CNAME records can be created for www.example.com, newproduct.example.com etc. • If a CNAME record is created for a subdomain, any other resource record sets for that subdomain cannot be created for e.g. if a CNAME created for www.example.com, not other resource record sets for which the value of the Name field is www.example.com can be created
Pointer (PTR)	Maps an IP address to a DNS name. **Used as a Reverse DNS lookup.**
MX (Mail Xchange)	**Specifies a mail exchange server**. Contains a decimal number that represents the priority of the MX record, and the domain name of an email server
NS (Name	**The value for an NS record is the domain name of**

Server)	**a name server.** An NS record identifies the name servers for the hosted zone.
SPF (Sender Policy Framework)	**SPF is a TXT record which is used as an email validation mechanism**. It allows DNS owners to authorize other IP addresses to send the emails from the domain. (example: if you are planning to send emails using “MailChimp” from your domain you will configure the SPF record to match MailChimp IP etc). • SPF records were formerly used to verify the identity of the sender of email messages, however is not recommended • Instead of an SPF record, a TXT record that contains the applicable value is recommended
SRV	An SRV record Value element consists of four space-separated values. The first three values are decimal numbers representing priority, weight, and port. The fourth value is a domain name for e.g. 10 5 80 hostname.example.com
TXT (Text)	Holds some text information. Can put any text. • A TXT record contains a space-separated list of double-quoted strings. A single string include a maximum of 255 characters. In addition to the characters that are permitted unescaped in domain names, space os allowed in TXT strings
SOA (Start of Authority)	• **SOA record provides information about a domain and the corresponding Amazon Route 53 hosted zone** • Information stored in the DNS. • Stores information about the name of the server which supplied the data. : o The name of the DNS server. o The administrator of the zone. o The current version of the data file. o The number of seconds that the **secondary** name server should wait, before checking for updates. o The number of seconds a **secondary** name server should wait before retrying a failed zone transfer. o The maximum number of seconds that a secondary name server can use data before it must either be refreshed or expire. o Default number of seconds for TTL file on resource records.

Alias resource record sets

- Route 53 supports alias resource record sets, which enables routing of queries to a CloudFront distribution, Elastic Beanstalk, ELB, an S3 bucket configured as a static website, or another Route 53 resource record set
- Alias records are not standard for DNS RFC and are an Route 53 extension to DNS functionality

- Alias records help map the apex zone (root domain without the www) records to the load balancer DNS name as the DNS specification requires “zone apex” to point to an ‘A’ record (ip address) and not to an CNAME
- Route 53 automatically recognizes changes in the resource record sets that the alias resource record set refers to for e.g. for a site pointing to an load balancer, if the ip of the load balancer changes, Route 53 will reflect those changes automatically in the DNS answers without any changes to the hosted zone that contains resource record sets
- If an alias resource record set points to a CloudFront distribution, a load balancer, or an S3 bucket, the time to live (TTL) can’t be set; Route 53 uses the CloudFront, load balancer, or Amazon S3 TTLs.

- **Hosted Zone**
  - **Collection of resource records set hosted by Amazon for a certain domain.** I

- **A hosted zone can be private or public:**

  - **Private**: Used when you want a host to be routed internally in an AWS VPC.

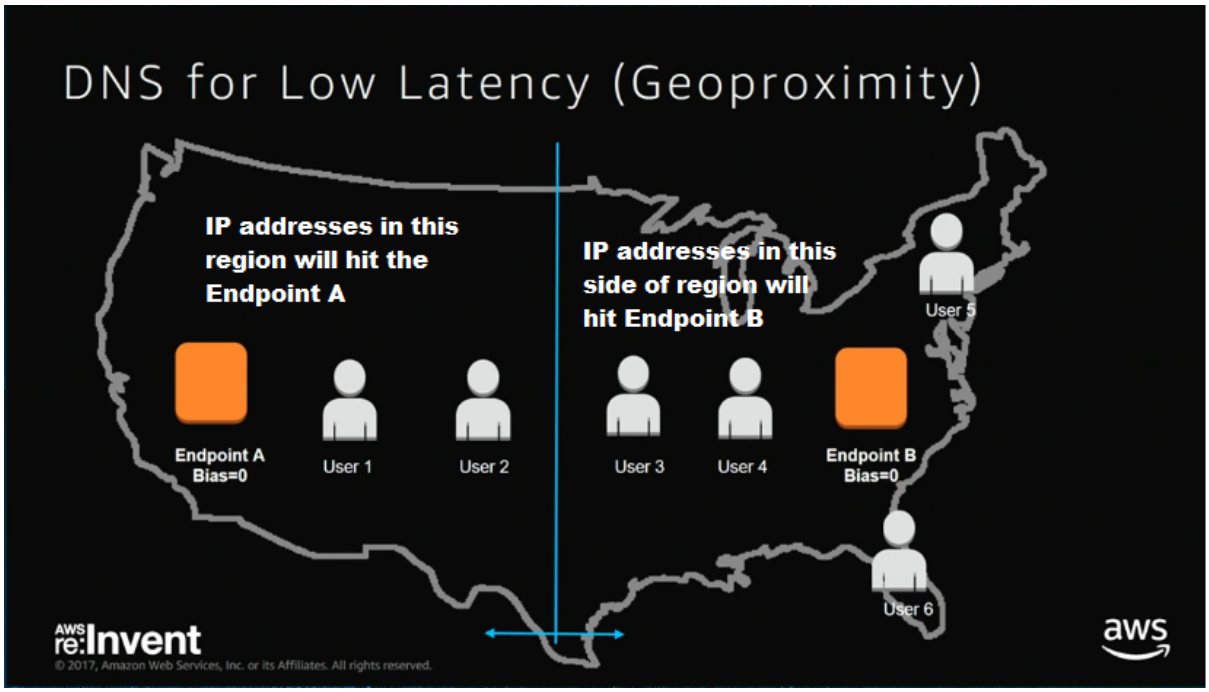
  - **Public**: Used when the host is on the internet.

- Routing Policies:

Routing Policies is a special feature of Route53 which lets you determine how Route53 responds to queries:

- **Simple routing policy**
  - Default policy. Ideally good when you have a single resource.
- **Fail-over routing policy**
  - Routes traffic to healthy resource when then the first resource is unhealthy (active-passive fail-over mechanism) . Cannot be created for private hosted zones.
- **Geolocation Routing**
  - Route traffic based on the geographic location where the traffic is originated. If Route 53 cannot identify the location of the IP address it routes it to the default resource set.
- **Geoproximity** – Route traffic based on the geographic location of your users and your resources:

•



- **Latency-based Routing**
  - Routes traffic based on the latency. Example: If you have created an ELB in the US East and one in China. When a user from Singapore visits your website, Route53 calculates the latency between Singapore and US East, Singapore and China. It will route the user to the latency endpoint.
- **Multivalue answer routing policy (new feature)**
  - AWS Route53 respond to DNS queries with up to eight healthy records selected at random
- **Weighted routing policy**
  - When you assign weight to each endpoint.
    - Use cases: 1) when you have the same application across multiple load balancer or EC2 instances but the instance type of EC2 instances are different. So, you assign a higher weight to the higher performance instance.
    - 2) To test new version of website (send 20% to the test environment , rest to the production)
    -

Route53 Key terms:

Route 53 is AWS DNS service, Domain purchase and health checking platform.

- **Top-level Domains (TLD)**

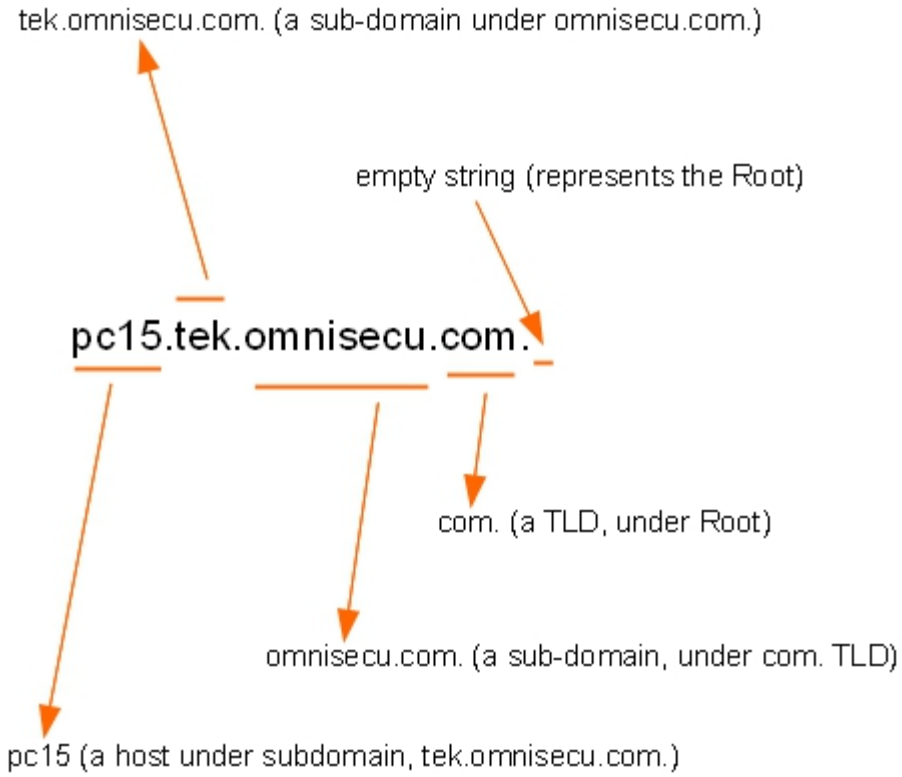
Last part of a domain name. example: .com, .org, net.
ICAN is responsible for it.

- **Subdomains**

- Godaddy.com, bingo.com are actually subdomain of .com but generally speaking we consider it as a domain.
- A subdomain can have multiple subdomains.
  - Blog.personal.wiki.com

- **Fully Qualified Domain Name (FQDN)**

FQDN is when the domain name is completely specified. It is also referred as an absolute domain name.



CloudFront

- CloudFront is a web service that speeds up distribution of static, dynamic web or streaming content to end users.
- CloudFront delivers the content through a worldwide network of data centers called **edge locations**
- CloudFront gives businesses and web application developers an easy and cost effective way **to distribute content** with **low latency and high data transfer speeds**
- **CloudFront speeds up the distribution of the content by routing each user request to the edge location that can best serve the content thus providing the lowest latency.**
- CloudFront dramatically reduces the number of network hops that users' requests must pass through, which helps improves performance, provide lower latency and higher data transfer rates.
- **CloudFront is a good choice for distribution of frequently accessed static content that benefits from edge delivery – like popular website images, videos, media files or software downloads**

Most Important Topic for Solution Architect Professional Exam

CloudFront Benefits

- CloudFront eliminates the expense and complexity of operating a network of cache servers in multiple sites across the internet and eliminates the need to over-provision capacity in order to serve potential spikes in traffic
- CloudFront also provides increased reliability and availability because copies of objects are held in multiple edge locations around the world
- CloudFront keeps persistent connections with the origin servers so that those files can be fetched from the origin servers as quickly as possible.
- CloudFront also uses techniques such as collapsing simultaneous viewer requests at an edge location for the same file into a single request to the origin server reducing the load on the origin
- **CloudFront integrates with AWS WAF, a web application firewall that helps protect web applications from attacks by allowing rules configured based on IP addresses, HTTP headers, and custom URI strings**
- Configuration & Content Delivery
- Configuration
    1. Origin servers need to be configured to get the files for distribution. An origin server stores the original, definitive version of the objects and can be a AWS hosted service for e.g. S3, EC2 or an on premise server
    2. Files (also called objects) can be added/uploaded to the Origin servers with public read permissions or permissions restricted to OAI
    3. Create a CloudFront distribution, which tells CloudFront which origin servers to get the files from when users request the files
    4. CloudFront sends the distribution configuration to all the edge locations

5. Website can be used with the CloudFront provided domain name or a custom alternate domain name
6. Origin server can be configured to limit access protocols, caching behaviour, add headers to the files to add TTL or the expiration time

- Content delivery to Users
  1. When user access the website, file or the object the DNS routes the request to the CloudFront edge location that can best serve the user's request with the lowest latency
  2. CloudFront returns the object immediately, if the requested object is present in the cache at the Edge location
  3. If the requested object does not exist in the cache at the edge location, CloudFront requests the object from the Origin server and returns it to the user as soon as it starts receiving it
  4. When the object reaches it expiration time, for any new request CloudFront checks with the Origin server for any latest versions, if it has the latest it uses the same object. If the Origin server has the latest version the same is retrieved, served to the user and cached as well

• Delivery Methods

Web distributions

- supports both static and dynamic content for e.g. html, css, js, images etc using HTTP or HTTPS.
- supports multimedia content on demand using progressive download and Apple HTTP Live Streaming (HLS).
- supports a live event, such as a meeting, conference, or concert, in real time. For live streaming, distribution can be created automatically using an AWS CloudFormation stack.
- origin servers can be either an Amazon S3 bucket or an HTTP server, for e.g., a web server or an AWS ELB etc

RMTP distributions

- s**upports streaming of media files using Adobe Media Server and the Adobe Real-Time Messaging Protocol (RTMP)**
- must use an S3 bucket as the origin.
- To stream media files using CloudFront, two types of files are needed
  - Media files
  - Media player for e.g. JW Player, Flowplayer, or Adobe flash
- End users view media files using the media player that is provided; not the locally installed on the computer of the device
- When an end user streams the media file, the media player begins to play the file content while the file is still being downloaded from CloudFront.
- Media file is not stored locally on the end user's system.
- Two CloudFront distributions are required, Web distribution for media Player and RMTP distribution for media files
- Media player and Media files can be stored in same origin S3 bucket or different buckets

Origin

- Each origin is either an S3 bucket or an HTTP server, for e.g., a web server, which stores the original content
- For HTTP server as the origin, the domain name of the resource needs to be mapped and files must be publicly readable
- For S3 bucket, use the bucket url or the static website endpoint url and the files either need to be publicly readable or secured using OAI
- Origin restrict access, for S3 only, can be configured using Origin Access Identity to prevent direct access to the S3 objects
- Distribution can have multiple origins for each bucket with one or more cache behaviours that route requests to each origin. Path pattern in a cache behaviour determines which requests are routed to the origin (S3 bucket) that is associated with that cache behaviour

Cache Behaviour

- Cache behaviour allows you to define
    - Path patterns to apply for the request. A default (*) pattern is created and multiple cache distributions can be added with patterns to take priority over the default path
    -

Viewer Protocol Policy

- Viewer Protocol policy can be configured to define the access protocol allowed. Can be either HTTP and HTTPS, or HTTPS only or HTTP redirected to HTTPS

HTTPS Connection

- Between CloudFront & Viewers, cache distribution can be configured to either allow HTTP or HTTPS requests, or use HTTPS only, or redirect all HTTP request to HTTPS
- Between CloudFront & Origin, cache distribution can be configured to require that CloudFront fetches objects from the origin by using HTTPS or CloudFront uses the protocol that the viewer used to request the objects.
- For S3 as origin,
    - for website, the protocol has to be HTTP as HTTPS is not supported
    - for S3 bucket, the default Origin protocol policy is Match Viewer and cannot be changed. So When CloudFront is configured to require HTTPS between the viewer and CloudFront, it automatically uses HTTPS to communicate with S3.
- CloudFront can also be configured to work with HTTPS for alternate domain names by using:-
    - Serving HTTPS Requests Using Dedicated IP Addresses
        - CloudFront associates the alternate domain name with a dedicated IP address, and the certificate is associated with the IP address. when a request is received from a DNS server for the IP address,
        - CloudFront uses the IP address to identify the distribution and the SSL/TLS certificate to return to the viewer
        - This method works for every HTTPS request, regardless of the browser or other viewer that the user is using.

        - Additional monthly charge (of about $600/month) is incurred for using dedicated IP address

    - Serving HTTPS Requests Using SNI

        - SNI custom SSL relies on the SNI extension of the TLS protocol, which allows multiple domains to be served over the same IP address by including the hostname, viewers are trying to connect to
        - With SNI method, CloudFront associates an IP address with the alternate domain name, but the IP address is not dedicated
        - CloudFront can’t determine, based on the IP address, which domain the request is for as the IP address is not dedicated
        - Browsers that support SNI automatically gets the domain name from the request URL & adds it to a new field in the request header.
        - When CloudFront receives an HTTPS request from a browser that supports SNI, it finds the domain name in the request header and responds to the request with the applicable SSL/TLS certificate.
        - Viewer and CloudFront perform SSL negotiation, and CloudFront returns the requested content to the viewer.
        - Older browsers do not support it
        - SNI Custom SSL is available at no additional cost beyond standard CloudFront data transfer and request fees

    - For End-to-End HTTPS connections certificate needs to be applied both between the Viewers and CloudFront & CloudFront and Origin, with the following requirements

        - HTTPS between viewers and CloudFront

            - Certificate that was issued by a trusted certificate authority (CA) such as Comodo, DigiCert, or Symantec;
            - Certificate provided by AWS Certificate Manager (ACM);
            - Self-signed certificate.

        - HTTPS between CloudFront and a custom origin

            - If the origin is not an ELB load balancer, the certificate must be issued by a trusted CA such as Comodo, DigiCert, or Symantec.
            - For ELB load balancer, certificate provided by ACM can be used

Allowed HTTP methods

- CloudFront supports GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE to get, add, update, and delete objects, and to get object headers.

- GET, HEAD, OPTIONS methods to use CloudFront only to get objects, object headers or retrieve a list of the options supported from your origin
- POST, PUT operations can also be performed for e.g. submitting data from a web form, which are directly proxied back to the Origin server
- CloudFront only caches responses to GET and HEAD requests and, optionally, OPTIONS requests. CloudFront does not cache responses to PUT, POST, PATCH, DELETE request methods and these requests are directed to the origin
- PUT, POST http methods also help for accelerated content uploads, as these operations will be sent to the origin e.g. S3 via the CloudFront edge location, improving efficiency, reducing latency, and allowing the application to benefit from the monitored, persistent connections that CloudFront maintains from the edge locations to the origin servers.
-

Improving CloudFront Edge Caches

- **Control the cache max-age**
  - To increase the cache hit ratio, origin can be configured to add a Cache-Control max-age directive to the objects.
  - Longer the interval less frequently it would be retrieved from the origin

- **Caching Based on Query String Parameters**
  - For Web distributions, CloudFront can be configured to cache based on the query parameters
  - Caching performance can be improved by
    - Configure CloudFront to forward only the query strings for which your origin will return unique objects.
    - Using the same case for the parameters values for e.g. parameter value A or a, CloudFront would cache the same request twice even if the response or object returned is identical
    - Using the same parameter order for e.g. for request a=x&b=y and b=y&a=x, CloudFront would cache the same request twice even thought the response or object returned is identical
  - For RTMP distributions, when CloudFront requests an object from the origin server, it removes any query string parameters.

- **Caching Based on Cookie Values**
  - For Web distributions, CloudFront can be configured to cache based on cookie values.
  - By default, it doesn’t consider cookies while caching on edge locations
  - Caching performance can be improved by
    - Configure CloudFront to forward only specified cookies instead of forwarding all cookies for e.g. if the request has 2 cookies with 3 possible values, CloudFront would cache all possible combinations even if the response takes into account a single cookie

        - Cookie names and values are both case sensitive so better to stick with the same case
        - Create separate cache behaviors for static and dynamic content, and configure CloudFront to forward cookies to the origin only for dynamic content for e.g. for css files, the cookies do not make sense as the object does not change with the cookie value
        - If possible, create separate cache behaviors for dynamic content for which cookie values are unique for each user (such as a user ID) and dynamic content that varies based on a smaller number of unique values reducing the number of combinations
    - For RTMP distributions, CloudFront cannot be configured to process cookies. When CloudFront requests an object from the origin server, it removes any cookies before forwarding the request to your origin. If your origin returns any cookies along with the object, CloudFront removes them before returning the object to the viewer.

- **Caching Based on Request Headers**
    - CloudFront can be configured to cache based on request headers
    - By default, CloudFront doesn't consider headers when caching your objects in edge locations.
    - CloudFront configured to cache based on request headers, does not change the headers that CloudFront forwards, only whether CloudFront caches objects based on the header values.
    - Caching performance can be improved by
        - Configure CloudFront to forward and cache based only specified headers instead of forwarding and caching based on all headers.
        - Try to avoid caching based on request headers that have large numbers of unique values.
        - CloudFront configured to forward all headers to your origin, CloudFront doesn't cache the objects associated with this cache behavior. Instead, it sends every request to the origin
        - CloudFront caches based on header values, it doesn't consider the case of the header name, but considers the case of the header value
    - For RTMP distributions, CloudFront cannot be configured to cache based on header values.

Object Caching & Expiration

- Object expiration determines how long the objects stay in a CloudFront cache before it fetches it again from Origin
- Low expiration time helps serve content that changes frequently and high expiration time helps improve performance and reduce load on the origin
- After expiration time, CloudFront checks if it still has the latest version
    - if the cache already has the latest version, the origin returns a 304 status code (Not Modified).
    - if the CloudFront cache does not have the latest version, the origin returns a 200 status code (OK) and the latest version of the object

- If an object in an edge location isn't frequently requested, CloudFront might evict the object, remove the object before its expiration date, to make room for objects that have been requested more recently.
- By default, each object automatically expires after 24 hours
- For Web distributions, the default behavior can be changed by
    - for the entire path pattern, cache behavior can be configured by setting of Minimum TTL, Maximum TTL and Default TTL values
    - for individual objects, origin can be configured to add a Cache-Control max-age or Cache-Control s-maxage directive, or an Expires header field to the object.
    - AWS recommends using Cache-Control max-age directive over Expires header to control object caching behavior
    - CloudFront uses only the value of Cache-Control max-age, if both the Cache-Control max-age directive and Expires header are specified
    - HTTP Cache-Control or Pragma header fields in a GET request from a viewer can't be used to force CloudFront to go back to the origin server for the object
    - By default, when the origin returns an HTTP 4xx or 5xx status code, CloudFront caches these error responses for five minutes and then submits the next request for the object to the origin to see whether the requested object is available and the problem has been resolved
- For RTMP distributions
    - Cache-Control or Expires headers can be added to objects to change the amount of time that CloudFront keeps objects in edge caches before it forwards another request to the origin.
    - Minimum duration is 3600 seconds (one hour). If you specify a lower value, CloudFront uses 3600 seconds.

Restrict viewer access

Serving Private Content

- To securely serve private content using CloudFront:

    - Require the users to access the private content by using special CloudFront signed URLs or signed cookies with following restrictions
        - an end date and time, after which the URL is no longer valid
        - start date time, when the URL becomes valid
        - ip address or range of addresses to access the URLs

    - Require that users access the S3 content only using CloudFront URLs, not S3 URLs. Requiring CloudFront URLs isn't required, but recommended to prevent users from bypassing the restrictions specified in signed URLs or signed cookies.
- Signed URLs or Signed Cookies can used with CloudFront using HTTP server as an origin. It requires the content to be publicly accessible and care should be taken to not share the direct URL of the content

- Restriction for Origin can be applied by

    - For S3, using Origin Access Identity to grant only CloudFront access using Bucket policies or Object ACL, to the content and removing any other access permissions
    - For HTTP server, custom header can be added by CloudFront which can be used at Origin to verify the request has come from CloudFront

- Trusted Signer
    - To create signed URLs or signed cookies, at least one AWS account (trusted signer) is needed that has an active CloudFront key pair
    - Once AWS account is added as trusted signer to the distribution, CloudFront starts to require that users use signed URLs or signed cookies to access the objects.
    - Private key from the trusted signer's key pair to sign a portion of the URL or the cookie. When someone requests a restricted object, CloudFront compares the signed portion of the URL or cookie with the unsigned portion to verify that the URL or cookie hasn't been tampered with. CloudFront also validates the URL or cookie is valid for e.g, that the expiration date and time hasn't passed.
    - Each Trusted signer AWS accounts used to create CloudFront signed URLs or signed cookies must have its own active CloudFront key pair, which should be frequently rotated
    - A maximum of 5 trusted signers can be assigned for each cache behavior or RTMP distribution

Signed URLs vs Signed Cookies

- **CloudFront signed URLs and signed cookies helps to secure the content and provide control to decide who can access the content**

- **Use signed URLs in the following cases:**
    - for RTMP distribution as signed cookies aren't supported
    - to restrict access to individual files, for e.g., an installation download for your application.
    - users using a client, for e.g. a custom HTTP client, that doesn't support cookies
- **Use signed cookies in the following cases:**
    - provide access to multiple restricted files, for e.g., all of the video files in HLS format or all of the files in the subscribers' area of a website.
    - don't want to change the current URLs.

Canned Policy vs Custom Policy

- Canned policy or a custom policy is a policy statement, used by the Signed URLs, helps define the restrictions for e.g. expiration date and time

Description	Canned Policy	Custom Policy
You can reuse the policy statement for multiple objects. To reuse the policy statement, you must use wildcard characters in the `Resource` object. For more information, see Values that You Specify in the Policy Statement for a Signed URL That Uses a Custom Policy (p. 191).)	No	Yes
You can specify the date and time that users can begin to access your content.	No	Yes (optional)
You can specify the date and time that users can no longer access your content.	Yes	Yes
You can specify the IP address or range of IP addresses of the users who can access your content.	No	Yes (optional)
The signed URL includes a base64-encoded version of the policy, which results in a longer URL.	No	Yes

- CloudFront validates the expiration time at the start of the event.
- If user is downloading a large object, and the url expires the download would still continue and the same for RTMP distribution.
- However, if the user is using range GET requests, or while streaming video skips to another position which might trigger an other event, the request would fail.

Serving Compressed Files

- **CloudFront can be configured to automatically compress files of certain types and serve the compressed files when viewer requests include Accept-Encoding: gzip in the request header**
- Compressing content, downloads are faster because the files are smaller as well as less expensive as the cost of CloudFront data transfer is based on the total amount of data served
- If serving from a custom origin, it can be used to
    - configure to compress files with or without CloudFront compression
    - compress file types that CloudFront doesn’t compress.
- If the origin returns a compressed file, CloudFront detects compression by the Content-Encoding header value and doesn’t compress the file again.
- CloudFront serves content using compression as below
    - CloudFront distribution is created and configured to compress content.
    - A viewer requests a compressed file by adding the Accept-Encoding: gzip header to the request.
    - At the edge location, CloudFront checks the cache for a compressed version of the file that is referenced in the request.
    - If the compressed file is already in the cache, CloudFront returns the file to the viewer and skips the remaining steps.

- If the compressed file is not in the cache, CloudFront forwards the request to the origin server (S3 bucket or a custom origin)
- Even if CloudFront has an uncompressed version of the file in the cache, it still forwards a request to the origin.
- Origin server returns an uncompressed version of the requested file
- CloudFront determines whether the file is compressible:
    1. file must be of a type that CloudFront compresses.
    2. file size must be between 1,000 and 10,000,000 bytes.
    3. response must include a Content-Length header for CloudFront to determine the size within valid compression limits. If the Content-Length header is missing, CloudFront won't compress the file.
    4. value of the Content-Encoding header on the file must not be gzip i.e. the origin has already compressed the file.
- If the file is compressible, CloudFront compresses it, returns the compressed file to the viewer, and adds it to the cache.
- The viewer uncompresses the file.

Distribution Details

Price Class

- CloudFront has edge locations all over the world and as cost for each edge location varies and the price charged for serving the requests also varies
- CloudFront edge locations are grouped into geographic regions, and regions have been grouped into price classes
    - **Default Price Class** – includes all the regions
    - **Another price class includes most regions** (the United States; Europe; Hong Kong, Korea, and Singapore; Japan; and India regions) but excludes the most-expensive regions
    - **A third price class includes only the least-expensive regions (the United States and Europe regions)**
- Price class can be selected to lower the cost but this would come only at the expense of performance (higher latency), as CloudFront would serve requests only from the selected price class edge locations
- CloudFront may, sometimes, serve request from a region not included within the price class, however you would be charged the rate for the least-expensive region in your selected price class

Alternate Domain Names (CNAMEs)

- CloudFront by default assigns a domain name for the distribution for e.g. d111111abcdef8.cloudfront.net
- An alternate domain name, also known as a CNAME, can be used to use own custom domain name for links to objects
- Both web and RTMP distributions support alternate domain names.
- CloudFront supports * wildcard at the beginning of a domain name instead of specifying subdomains individually.
- However, wildcard cannot replace part of a subdomain name for e.g. *domain.example.com, or cannot replace a subdomain in the middle of a domain name for e.g. subdomain.*.example.com.

Geo Restriction (Geoblocking)

- Geo restriction can help allow or prevent users in selected countries from accessing the content,
- CloudFront distribution can be configured either to allow users in
    - whitelist of specified countries to access the content or to
    - deny users in a blacklist of specified countries to access the content
- Geo restriction can be used to restrict access to all of the files that are associated with a distribution and to restrict access at the country level
- CloudFront responds to a request from a viewer in a restricted country with an HTTP status code 403 (Forbidden)
- Use a third-party geolocation service, if access is to be restricted to a subset of the files that are associated with a distribution or to restrict access at a finer granularity than the country level

CloudFront with Amazon S3

- **CloudFront can be used to distribute the content from an S3 bucket**
- **For an RTMP distribution, S3 bucket is the only supported origin and custom origins cannot be used**
- Using CloudFront over S3 has the following benefits
    - can be more cost effective if the objects are frequently accessed as at higher usage, the price for CloudFront data transfer is much lower than the price for S3 data transfer.
    - downloads are faster with CloudFront than with S3 alone because the objects are stored closer to the users
- When using S3 as the origin for a distribution and the bucket is moved to a different region, CloudFront can take up to an hour to update its records to include the change of region when both of the following are true:
    - Origin Access Identity (OAI) is used to restrict access to the bucket
    - Bucket is moved to an S3 region that requires Signature Version 4 for authentication

Origin Access Identity

- **With S3 as origin, objects in S3 must be granted public read permissions and hence the objects are accessible from both S3 as well as CloudFront**
- Even though, CloudFront does not expose the underlying S3 url, it can be known to the user if shared directly or used by applications
- For using CloudFront signed URLs or signed cookies to provide access to the objects, it would be necessary to prevent users from having direct access to the S3 objects
- Users accessing S3 objects directly would
    - bypass the controls provided by CloudFront signed URLs or signed cookies, for e.g., control over the date time that a user can no longer access the content and the IP addresses can be used to access content
    - CloudFront access logs are less useful because they’re incomplete.
- Origin Access Identity (OAI) can be used to prevent users from directly accessing objects from S3
- Origin access identity, which is a special CloudFront user, can be created and associated with the distribution.

- S3 bucket/object permissions needs to be configured to only provide access to the Origin Access Identity
- When users access the object from CloudFront, it uses the OAI to fetch the content on users behalf, while direct access to the S3 objects is restricted

Working with Objects

- **CloudFront can be configured to include custom headers or modify existing headers whenever it forwards a request to the origin, to**
    - validate the user is not accessing the origin directly, bypassing CDN
    - identify the CDN from which the request was forwarded, if more than one CloudFront distribution is configured to use the same origin
    - if users use viewers that don't support CORS, configure CloudFront to forward the Origin header to the origin. That will cause the origin to return the Access-Control-Allow-Origin header for every request

Adding & Updating Objects

- **Objects just need to be added to the Origin and CloudFront would start distributing them when accessed**
- Objects served by CloudFront the Origin, can be updated either by
    - Overwriting the Original object
    - Create a different version and updating the links exposed to the user
- For updating objects, its recommended to use versioning for e.g. have files or the entire folders with versions, so the the links can be changed when the objects are updated forcing a refresh
- With versioning,
    - there is no time wait for an object to expire before CloudFront begins to serve a new version of it
    - there is no difference in consistency in the object served from the edge
    - no cost involved to pay for object invalidation.

Removing/Invalidating Objects

- **Objects, by default, would be removed upon expiry (TTL) and the latest object would be fetched from the Origin**
- Objects can also be removed from the edge cache before it expires
    - Change object name (versioning) to serve a different version of the object that has a different name
    - Invalidate the object from edge caches. For the next request, CloudFront returns to the Origin to fetch the object
- For Web distributions,
    - If your objects need to be updated frequently, changing Object name (Versioning) is recommended over Invalidating objects, as it
        - enables to control which object a request returns even when the user has a version cached either locally or behind a corporate caching proxy. If an object is invalidated, the user might continue to see the old version until it expires from those caches.

        - makes it easier to analyze the results of object changes as CloudFront access logs include the names of the objects
        - provides a way to serve different versions to different users.
        - simplifies rolling forward & back between object revisions.
        - is less expensive, as no charges for invalidating objects.
        - for e.g. change header-v1.jpg to header-v2.jpg
    - Invalidating objects from the cache
        - objects in the cache can be invalidated explicitly before they expire to force a refresh
        - allows to invalidate selected objects
        - allows to invalidate multiple objects for e.g. objects in a directory or all of the objects whose names begin with the same characters, you can include the * wildcard at the end of the invalidation path.
        - A specified number of invalidation paths can be submitted each month for free. Any invalidation requests more than the allotted no. per month, fee is charged for each submitted invalidation path
        - First 1,000 invalidation paths requests submitted per month are free; charges apply for each invalidation path over 1,000 in a month.
        - Invalidation path can be for a single object for e.g. /js/ab.js or for multiple objects for e.g. /js/* and is counted as a single request even if the * wildcard request may invalidate thousands of objects
- For RTMP distribution, objects served cannot be invalidated

Partial Requests (Range GETs)

- **Partial requests using Range headers in a GET request helps to download the object in smaller units, improving the efficiency of partial downloads and the recovery from partially failed transfers.**
- For a partial GET range request, CloudFront
    - checks the cache in the edge location for the requested range or the entire object and if exists, serves it immediately
    - if the requested range does not exist, it forwards the request to the origin and may request a larger range than the client requested to optimize performance
    - if the origin supports range header, it returns the requested object range and CloudFront returns the same to the viewer
    - if the origin does not support range header, it returns the complete object and CloudFront serves the entire object and caches it for future.
    - CloudFront uses the cached entire object to serve any future range GET header requests

Access Logs

- **CloudFront can be configured to create log files that contain detailed information about every user request that CloudFront receives.**
- **Access logs are available for both web and RTMP distributions.**

- **With logging enabled, an S3 bucket can be specified where CloudFront would save the files**
- CloudFront delivers access logs for a distribution periodically, up to several times an hour
- CloudFront usually delivers the log file for that time period to the S3 bucket within an hour of the events that appear in the log. Note, however, that some or all log file entries for a time period can sometimes be delayed by up to 24 hours

- CloudFront Cost
- **CloudFront charges are based on actual usage of the service in four areas:**
    - **Data Transfer Out to Internet**
        - charges are applied for the volume of data transferred out of the CloudFront edge locations, measured in GB
        - Data transfer out from AWS origin (e.g., S3, EC2, etc.) to CloudFront are no longer charged. This applies to data transfer from all AWS regions to all global CloudFront edge locations
    - **HTTP/HTTPS Requests**
        - number of HTTP/HTTPS requests made for the content
    - **Invalidation Requests**
        - per path in the invalidation request
        - A path listed in the invalidation request represents the URL (or multiple URLs if the path contains a wildcard character) of the object you want to invalidate from CloudFront cache
    - **Dedicated IP Custom SSL certificates associated with a CloudFront distribution**
        - $600 per month for each custom SSL certificate associated with one or more CloudFront distributions using the Dedicated IP version of custom SSL certificate support, pro-rated by the hour

- CloudFront - Key points to remember

- CDN of AWS. CloudFront (the location of the data center is the edge location, CloudFront is a CDN like Akamai)
- To reduce the distance between the user and the webserver, CDN stores the cached version of the content in various locations called “edge locations” . The user is routed to the nearest edge location.
- Cloudfront can work with AWS resources such as S3 or non AWS resources (websites not hosted on AWS)

- Distributions: the CDN domain name. In order to use CloudFront you would need to create a distribution d123123.cloudfront.net . All you need to do is replace your domain name with the distribution name.
- **Origins**: the location which CloudFront fetches files. CloudFront can fetch files from the below:
    - S3 Bucket.
    - Custom origin.
    - EC2 Instance.
    - ELB
        - Specify the URL of the load balancer for the domain name of your origin server

- **Cache Control**: the default expiry time is after 24 hours. This can be controlled by using the Cache-Control Header. To remove files from the cache the **invalidation** API needs to be called.
- Every CloudFront web distribution must be associated either with the default CloudFront certificate or with a custom SSL certificate.
    - Before you can delete an SSL certificate you need to either rotate SSL certificates or revert from using a custom SSL certificate to using the default Cloudfront certificate.
- **Optimizing Content Caching**
    - To set up and manage caching of objects to improve performance and meet your business requirements.
    - Main Topics:
        - Caching Content Based on Query String Parameters
            - Some web applications use query strings to send information to the origin. A query string is the part of a web request that appears after a ? character;
            - For web distributions, you can choose whether you want CloudFront to forward query strings to your origin and, if so, whether to cache your content based on all parameters or on selected parameters.
            - Suppose your website is available in five languages. The directory structure and file names for all five versions of the website are identical. As a user views your website, requests that are forwarded to CloudFront include a language query string parameter based on the language that the user chose. You can configure CloudFront to forward query strings to the origin and to cache based on the language parameter. If you configure your web server to return the version of a given page that corresponds with the selected language, CloudFront will cache each language version separately, based on the value of the language query string parameter.
            - In this example, if the main page for your website is main.html, the following five requests will cause CloudFront to cache main.html five times, once for each value of the language query string parameter:
            - http://d111111abcdef8.cloudfront.net/main.html?language=de
            - http://d111111abcdef8.cloudfront.net/main.html?language=en
            - http://d111111abcdef8.cloudfront.net/main.html?language=es
            - http://d111111abcdef8.cloudfront.net/main.html?language=fr
            - http://d111111abcdef8.cloudfront.net/main.html?language=jp
            - **RTMP distributions and some HTTP servers do not process query string parameters**

- **Optimizing Caching**

- Configure CloudFront to cache based on query string parameters helps to reduce the number of requests that CloudFront forwards to your origin, which reduces the load on your origin server and also reduces latency because more objects are served from CloudFront edge locations.

- **Caching Content Based on Request Headers**
  - For web distributions, CloudFront lets you choose whether you want CloudFront to forward headers to your origin and to cache separate versions of a specified object based on the header values in viewer requests. This allows you to serve different versions of your content based on the device the user is using, the location of the viewer, the language the viewer is using, and a variety of other criteria. F**or RTMP distributions, you cannot configure CloudFront to cache based on header values.**

  - **Configuring CloudFront to Cache Objects Based on the Device Type**
    - If you want CloudFront to cache different versions of your objects based on the device a user is using to view your content, configure CloudFront to forward the applicable headers to your custom origin:
      - CloudFront-Is-Desktop-Viewer
      - CloudFront-Is-Mobile-Viewer
      - CloudFront-Is-SmartTV-Viewer
      - CloudFront-Is-Tablet-Viewer
    - **Configuring CloudFront to Cache Objects Based on the Language of the Viewer**
      - If you want CloudFront to cache different versions of your objects based on the language specified in the request, program your application to include the language in the Accept-Language header, and configure CloudFront to forward the Accept-Language header to your origin.
    - **Configuring CloudFront to Cache Objects Based on the Location of the Viewer**
      - If you want CloudFront to cache different versions of your objects based on the country that the request came from, configure CloudFront to forward the CloudFront-Viewer-Country header to your origin. CloudFront automatically converts the IP address that the request came from into a two-letter country code. For an easy-to-use list of country codes, sortable by code and by country name, see the Wikipedia entry .
    - **Configuring CloudFront to Cache Objects Based on the Protocol of the Request**
      - If you want CloudFront to cache different versions of your objects based on the protocol of the request, HTTP or HTTPS, configure CloudFront to forward the CloudFront-Forwarded-Proto header to your origin.

- **Specifying a Default Root Object**

- You can configure CloudFront to return a specific object (the default root object) when a user requests the root URL for your web distribution instead of requesting an object in your distribution. Specifying a default root object lets you avoid exposing the contents of your distribution and show an Error page.

AWS ElastiCache

- Improves performance of web applications
- AWS ElastiCache is a managed web service that helps deploy and run Memcached or Redis protocol-compliant cache clusters in the cloud easily
- ElastiCache is available in two flavours: Memcached and Redis

- **ElastiCache helps**
  - simplify and offload the management, monitoring, and operation of in-memory cache environments, enabling the engineering resources to focus on developing applications
  - automate common administrative tasks required to operate a distributed cache environment.
  - i**mproves the performance of web applications by allowing retrieval of information from a fast, managed, in-memory caching system, instead of relying entirely on slower disk-based databases.**
  - **helps improve load &response times to user actions and queries, but also reduce the cost associated with scaling web applications**
  - helps automatically detect and replace failed cache nodes, providing a resilient system that mitigates the risk of overloaded databases, which can slow website and application load times
  - provides enhanced visibility into key performance metrics associated with the cache nodes through integration with CloudWatch

- **ElastiCache provides in-memory caching which can**
  - **significantly lower latency and improve throughput for many**
    - read-heavy application workloads for e.g. social networking, gaming, media sharing and Q&A portals or
    - compute-intensive workloads such as a recommendation engine
  - improve application performance by storing critical pieces of data in memory for low-latency access.
  - be used to cache results of I/O-intensive database queries or the results of computationally-intensive calculations.
- **ElastiCache currently allows access only from the EC2 network and cannot be accessed from outside networks like on-premises servers**

	Redis	Memcached
Persistence	Provides persistence storage and is a replacement for DB	Purely a caching solution and uses DB as the origin of the data
Object type	Complex data objects such as hashes, lists, sets etc.	Simple key value storage
Scaling	Vertical scaling supported. Horizontal Scaling not possible. Read Replicas can be created	Vertical and Horizontal Scaling supported
Multi-AZ	Multi-AZ supported & Automatic failover to the backup node	Multi-AZ not supported
Backup & Restore	Backup & Restore capabilities supported	Backup & Restore capabilities not supported
Pub/Sub capabilities	Pub/Sub capabilities provided	Pub/Sub capabilities not provided
Size	Values up to 512MB per key	Values up to 1MB per key

Redis

- is an open source, BSD licensed, advanced key-value cache & store
- ElastiCache enables the management, monitoring and operation of a Redis node; creation, deletion and modification of the node
- ElastiCache for Redis can be used as a primary in-memory key-value data store, providing fast, sub millisecond data performance, high availability and scalability up to 16 nodes plus up to 5 read replicas, each of up to 3.55 TiB of in-memory data
- **ElastiCache for Redis supports (similar to RDS features)**
  - **Redis Master/Slave replication.**
  - **Multi-AZ operation by creating read replicas in another AZ**
  - **Backup and Restore feature for persistence by snapshotting**
- **ElastiCache for Redis can be vertically scaled upwards by selecting a larger node type, however it cannot be scaled down**
- **Parameter group can be specified for Redis during installation, which acts as a "container" for Redis configuration values that can be applied to one or more Redis primary clusters**
- Append Only File (AOF)
  - provides persistence and can be enabled for recovery scenarios
  - if a node restarts or service crash, Redis will replay the updates from an AOF file, thereby recovering the data lost due to the restart or crash

    - cannot protect against all failure scenarios, cause if the underlying hardware fails, a new server would be provisioned and the AOF file will no longer be available to recover the data
    - Enabling Redis Multi-AZ as a Better Approach to Fault Tolerance, as failing over to a read replica is much faster than rebuilding the primary from an AOF file

Redis Read Replica

- Read Replicas help provide Read scaling and handling failures
- Read Replicas are kept in sync with the Primary node using Redis's asynchronous replication technology
- R**edis Read Replicas can help**
    - **Horizontal scaling beyond the compute or I/O capacity of a single primary node for read-heavy workloads.**
    - **Serving read traffic while the primary is unavailable either being down due to failure or maintenance**
    - **Data protection scenarios to promote a Read Replica as primary node, in case the primary node or the AZ of the primary node fails**
- ElastiCache supports initiated or forced failover where it flips the DNS record for the primary node to point at the read replica, which is in turn promoted to become the new primary
- Read replica cannot span across regions and may only be provisioned in the same or different AZ of the same Region as the cache node primary

Redis Multi-AZ

- ElastiCache for Redis shard consists of a primary and up to 5 read replicas
- Redis asynchronously replicates the data from the primary node to the read replicas
- **ElastiCache for Redis Multi-AZ mode**
    - **provides enhanced availability and smaller need for administration as the node failover is automatic**
    - **impact on the ability to read/write to the primary is limited to the time it takes for automatic failover to complete**
    - **no longer needs monitoring of Redis nodes and manually initiating a recovery in the event of a primary node disruption**
- During certain types of planned maintenance, or in the unlikely event of ElastiCache node failure or AZ failure,
    - it automatically detects the failure,
    - selects a replica, depending upon the read replica with the smallest asynchronous replication lag to the primary, and promotes it to become the new primary node
    - it will also propagate the DNS changes so that the the primary endpoint remains the same
- If Multi-AZ is not enabled,
    - ElastiCache monitors the primary node
    - in case the node becomes unavailable or unresponsive, it will repair the node by acquiring new service resources

- it propagates the DNS endpoint changes to redirect the node's existing DNS name to point to the new service resources.
- If the primary node cannot be healed and you will have the choice to promote one of the read replicas to be the new primary

Redis Backup & Restore

- Backup and Restore allows users to create snapshots of the Redis clusters
- Snapshots can be used for recovery, restoration, archiving purpose or warm start an ElastiCache for Redis cluster with preloaded data
- Snapshots can created on a cluster basis and uses Redis' native mechanism to create and store an RDB file as the snapshot
- Increased latencies for a brief period at the node might be encountered while taking a snapshot, and is recommended to be taken from a Read Replica minimizing performance impact
- Snapshots can be created either automatically (if configured) or manually
- ElastiCache for Redis cluster when deleted removes the automatic snapshots. However, manual snapshots are retained

Memcached

- is an in-memory key-value store for small chunks or arbitrary data
- ElastiCache for Memcached can be used to cache a variety of objects
  - from the content in persistent data stores such as RDS, DynamoDB, or self-managed databases hosted on EC2) to
  - dynamically generated web pages for e.g. with Nginx or
  - transient session data that may not require a persistent backing store
- **ElastiCache for Memcached**
  - **can be scaled Vertically by increasing the node type size**
  - **can be scaled Horizontally by adding and removing nodes**
  - **does not support persistence of data**
- **ElastiCache for Memcached cluster can have**
  - **nodes can span across multiple AZs within the same region**
  - **maximum of 20 nodes per cluster with a maximum of 100 nodes per region (soft limit and can be extended)**
- ElasticCache for Memcached supports auto discovery, which enables automatic discovery of cache nodes by clients when they are added to or removed from an ElastiCache cluster

ElastiCacheMitigating Failures

- ElastiCache should be designed to plan so that failures have a minimal impact upon your application and data

- **Mitigating Failures when Running Memcached**
  - **Mitigating Node Failures**
    - **spread the cached data over more nodes**
    - **as Memcached does not support replication, a node failure will always result in some data loss from the cluster**

        - **having more nodes will reduce the proportion of cache data lost**
    - Mitigating Availability Zone Failures
        - locate the nodes in as many availability zones as possible, only the data cached in that AZ is lost, not the data cached in the other AZs

- **Mitigating Failures when Running Redis**
    - **Mitigating Cluster Failures**
        - **Redis Append Only Files (AOF)**
            - **enable AOF so whenever data is written to the Redis cluster, a corresponding transaction record is written to a Redis AOF**
            - **when Redis process restarts, ElastiCache creates a replacement cluster and provisions it and repopulating it with data from AOF**
            - **It is time consuming**
            - **AOF can get big**
            - **Using AOF cannot protect you from all failure scenarios**
        - Redis Replication Groups
            - A Redis replication group is comprised of a single primary cluster which your application can both read from and write to, and from 1 to 5 read-only replica clusters.
            - Data written to the primary cluster is also asynchronously updated on the read replica clusters
            - When a Read Replica fails, ElastiCache detects the failure, replaces the instance in the same AZ and synchronizes with the Primary Cluster
            - Redis Multi-AZ with Automatic Failover, ElastiCache detects Primary cluster failure, promotes a read replica with least replication lag to primary
            - Multi-AZ with Auto Failover is disabled, ElastiCache detects Primary cluster failure, creates a new one and syncs the new Primary with one of the existing replicas
    - Mitigating Availability Zone Failures
        - locate the clusters in as many availability zones as possible

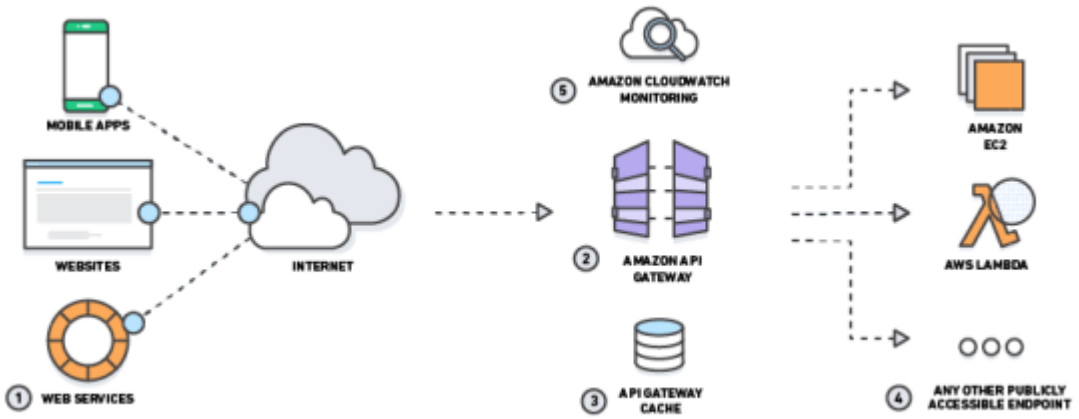
ElastiCache Key Notes

- You can manage HTTP session data from the web servers using an In-Memory Key/Value store such as Redis and Memcached. Redis is an open source, in-memory data structure store used as a database, cache, and message broker. Memcached is an in-memory key-value store for small arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.

- In AWS, you can use Amazon **ElastiCache** which offers fully managed Redis and Memcached service to manage and store session data for your web applications.

AWS API Gateway

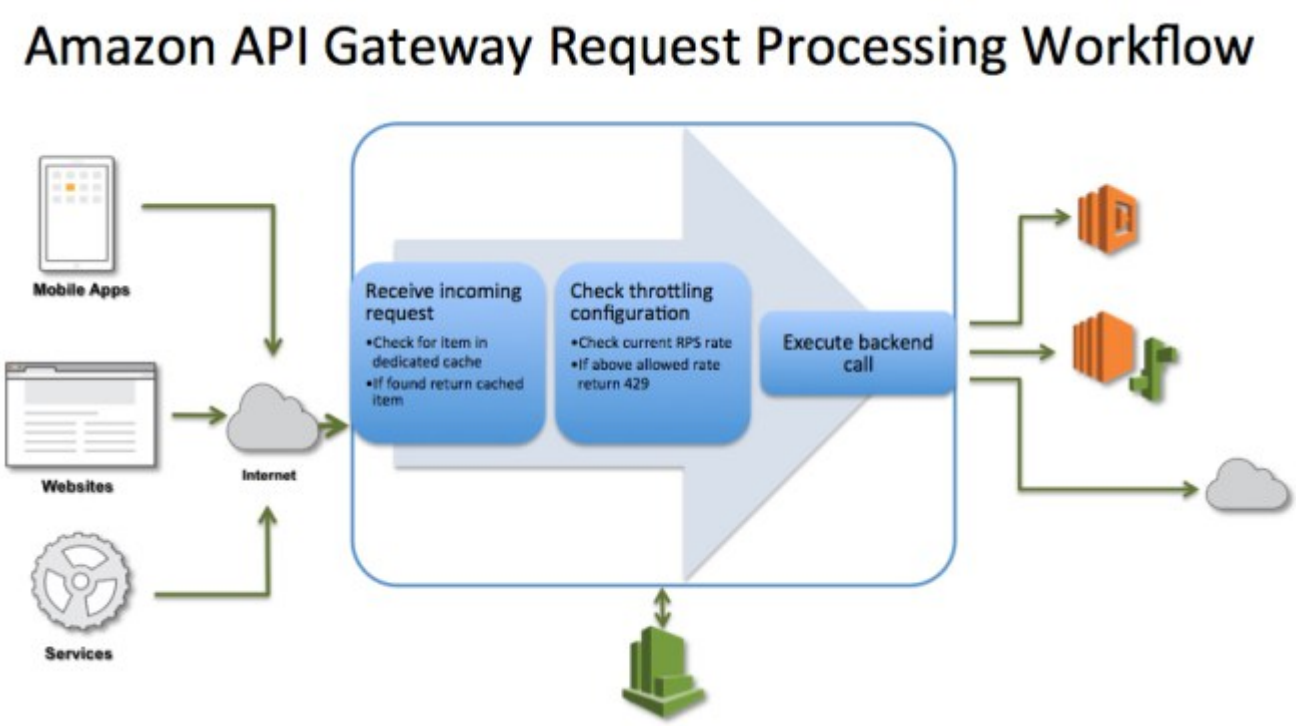
- **AWS API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale**
- **API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management.**
- API Gateway has no minimum fees or start-up costs and charges only for the API calls received and the amount of data transferred out.
- API Gateway acts as a proxy to the configured back-end operations.
- API Gateway scales automatically to handle the amount of traffic the API receives
- **API Gateway expose HTTPS endpoints only for all the APIs created. It does not support unencrypted (HTTP) endpoints**
- APIs built on API Gateway can accept any payloads sent over HTTP with typical data formats include JSON, XML, query string parameters, and request headers
- **API Gateway can communicate to multiple backends**
    - Lambda functions
    - AWS Step functions state machines
    - HTTP endpoints exposed through Elastic Beanstalk, ELB or EC2 servers
    - Non AWS hosted HTTP based operations accessible via public Internet
- API Gateway endpoints are always public to the Internet and does not run within an VPC. Proxy requests to backend operations also need to be publicly accessible on the Internet.



API Gateway helps with creating and managing APIs

- Metering
    - **automatically meters traffic to the APIs** and and lets you extract utilization data for each API key.
    - define plans that meter, restrict third-party developer access, configure throttling, and quota limits on a per API key basis
- Security
    - helps removing authorization concerns from the backend code
    - allows leveraging of AWS administration and security tools, such as IAM and Cognito, to authorize access to APIs
    - **can verify signed API calls on your behalf** using the same methodology AWS uses for its own APIs
    - **supports custom authorizers written as Lambda functions** and verify incoming bearer tokens
    - **automatically protects the backend systems from distributed denial-of-service (DDoS) attacks**, whether attacked with counterfeit requests (Layer 7) or SYN floods (Layer 3).
- Resiliency
    - helps manage traffic with throttling so that backend operations can withstand traffic spikes
    - **helps improve the performance of the APIs and the latency** end users experience by caching the output of API calls to avoid calling the backend every time.
- Operations Monitoring
    - **integrates with CloudWatch and provides a metrics dashboard to monitor calls to API services**
    - integrates with CloudWatch Logs to receive error, access or debug logs
    - provides with backend performance metrics covering API calls, latency data and error rates.
- Lifecycle Management
    - **allows multiple API versions and multiple stages (development, staging, production etc.)** for each version simultaneously so that existing applications can continue to call previous versions after new API versions are published.
    - saves the history of the deployments, which allows roll-back of a stage to a previous deployment at any point, using APIs or console
- Designed for Developers
    - **allows you to specify a mapping template to generate static content to be returned, helping you mock APIs before the backend is ready**
    - **helps reduce cross-team development effort and time-to-market** for applications and allow dependent teams to begin development while backend processes is still built

- API Gateway Throttling and Caching



- Throttling
  - API Gateway provides throttling at multiple levels including global and by service call and limits can be set for standard rates and bursts
  - It tracks the number of requests per second. Any requests over the limit will receive a 429 HTTP response
  - Throttling ensures that API traffic is controlled to help the backend services maintain performance and availability.
- Caching
  - API Gateway provides API result caching by provisioning an API Gateway cache and specifying its size in gigabytes
  - Caching helps improve performance and reduces the traffic sent to the back end
  - **API Gateway handles the request in the following manner**
    - If caching is not enabled and throttling limits have not been applied, then all requests pass through to the backend service until the account level throttling limits are reached.
    - If throttling limits specified, then API Gateway will shed necessary amount of requests and send only the defined limit to the back-end
    - If a cache is configured, then API Gateway will return a cached response for duplicate requests for a customizable time, but only if under configured throttling limits
- API Gateway does not arbitrarily limit or throttle invocations to the backend operations and all requests that are not intercepted by throttling and caching settings are sent to your backend operations.

AWS Lambda

- **AWS Lambda offers Server-less computing**
- Serverless computing allows applications and services to be built and run without thinking about servers. With server-less computing, application still runs on servers, but all the server management is done by AWS.
- Lambda lets you run code without provisioning or managing servers, where you pay only for the compute time when the code is running.
- Lambda is priced on a pay per use basis and there are no charges when the code is not running
- Lambda allows running of code for any type of application or backend service with zero administration
- Lambda performs all the operational and administrative activities on your behalf, including capacity provisioning, monitoring fleet health, applying security patches to the underlying compute resources, deploying code, running a web service front end, and monitoring and logging the code.
- **Lambda does not provide access to the underlying compute infrastructure**

- **Scalability and availability**
    - Lambda provides easy scaling and high availability to the code without additional effort on your part.
    - Lambda is designed to process events within milliseconds.
    - Latency will be higher immediately after a Lambda function is created, updated, or if it has not been used recently.
    - Lambda is designed to run many instances of the functions in parallel
    - Lambda is designed to use replication and redundancy to provide high availability for both the service and the Lambda functions it operates.
    - There are no maintenance windows or scheduled downtimes for either
    - For any Lambda function updates, there is a brief window of time, less then a minute, when requests would be served by both the versions
    - Lambda has a default safety throttle for number of concurrent executions per account per region

- **Security**

    - **Lambda stores code in S3 and encrypts it at rest and performs additional integrity checks while the code is in use.**
    - **Each AWS Lambda function runs in its own isolated environment, with its own resources and file system view**
- All calls made to AWS Lambda must complete execution within 300 seconds. Default timeout is 3 seconds. Timeout can be set the timeout to any value between 1 and 300 seconds.
- AWS Step Functions can help coordinate a series of Lambda functions in a specific order. Multiple Lambda functions can be invoked sequentially,

passing the output of one to the other, and/or in parallel, while the state is being maintain by Step Functions.
    - AWS X-Ray helps tracing for Lambda functions, which provides insights such as Lambda service overhead, function init time, and function execution time
- Lambda Functions & Event Sources

Core components of Lambda are Lambda functions and event sources.

- Event source is an AWS service or custom application that publishes events
- Lambda function is the custom code that processes the events

Lambda Functions

- Each Lambda function has associated configuration information, such as its name, description, entry point, and resource requirements
- Lambda may choose to retain an instance of the function and reuse it to serve a subsequent request, rather than creating a new copy
- Each Lambda function receives 500MB of non-persistent disk space in its own /tmp directory.
- Design Lambda function as stateless
    - **Lambda functions should be stateless, to allow AWS Lambda launch as many copies of the function as needed as per the demand**
    - Local file system access, child processes, and similar artefacts may not extend beyond the lifetime of the request
    - **State can be maintained externally in DynamoDB or S3**
- **Lambda function can be granted permissions to access other resources using an IAM role**
- **Lambda functions have the following restrictions**
    - **Inbound network connections are blocked by AWS Lambda**
    - **Outbound connections only TCP/IP sockets are supported**
    - **ptrace (debugging) system calls are blocked**
    - **TCP port 25 traffic is also blocked as an anti-spam measure.**
- **Lambda automatically monitors functions, reporting real-time metrics through CloudWatch, including total requests, latency, error rates, and throttled requests**
- Lambda automatically integrates with CloudWatch logs, creating a log group for each function and providing basic application life-cycle event log entries, including logging the resources consumed for each use of that function
- **Lambda functions supports code written in**
    - **Node.js (JavaScript)**
    - **Python**
    - **Java (Java 8 compatible)**
    - **C# (.NET Core)**
    - **Go**
- **Security**
    - For sensitive information, for e.g. passwords, AWS recommends using client-side encryption using AWS Key Management Service

and store the resulting values as ciphertext in your environment variable.
    - Lambda function code should include the logic to decrypt these values
- **Versioning**
    - **Each AWS Lambda function has a single, current version of the code and there is no versioning of the same function.**
    - **Each Lambda function version has a unique ARN and after it is published it is immutable**
    - Versioning can be implemented using .
    - Lambda supports creating aliases, which are mutable, for each Lambda function versions
    - Alias is a pointer to a specific function version, with unique ARN
    - Each alias maintains an ARN for a function version to which it points
    - An alias can only point to a function version, not to another alias
    - **Alias helps in rolling out new changes or rolling back to old versions**
- **Failure Handling**
    - For S3 bucket notifications and custom events, Lambda will attempt execution of the function three times in the event of an error condition in the code or if a service or resource limit is exceeded
    - For ordered event sources, for e.g. DynamoDB Streams and Kinesis streams, that Lambda polls, it will continue attempting execution in the event of a developer code error until the data expires.
    - Kinesis and DynamoDB Streams retain data for a minimum of 24 hours
    - Dead Letter Queues can be configured for events to be placed, once the retry policy for asynchronous invocations is exceeded

Lambda Event Sources

Lambda Execution Model

- When AWS Lambda executes the Lambda function, it takes care of provisioning and managing resources needed to run the Lambda function.
- When a Lambda function is invoked for the first time or after it has been updated there is a latency for bootstrapping as Lambda tries to reuse the Execution Context for subsequent invocations of the Lambda function
- When a Lambda function is invoked, Lambda launches an Execution Context based on the provided configuration settings i.e. memory and execution time
- After a Lambda function is executed, Lambda maintains the Execution Context for some time in anticipation subsequent function invocation
- Execution Context is a temporary runtime environment that initializes any external dependencies of the Lambda function code, for e.g. database connections or HTTP endpoints.
- Subsequent invocations perform better performance as there is no need to “cold-start” or initialize those external dependencies
- Lambda manages Execution Context creations and deletion, there is no AWS Lambda API to manage Execution Context.

- Lambda Best Practices
  - **Lambda function code should be stateless,** and ensure there is no affinity between the code and the underlying compute infrastructure.
  - Instantiate AWS clients outside the scope of the handler to take advantage of connection re-use.
  - Make sure you have set +rx permissions on your files in the uploaded ZIP to ensure Lambda can execute code on your behalf.
  - Lower costs and improve performance by minimizing the use of startup code not directly related to processing the current event.
  - Use the built-in CloudWatch monitoring of your Lambda functions to view and optimize request latencies.
  - Delete old Lambda functions that you are no longer using.

Lambda@Edge

- allows running of code across AWS locations globally without provisioning or managing servers, responding to end users at the lowest network latency
- Lambda function can be configured to be triggered in response to CloudFront requests, which includes
  - **Viewer Request** – event occurs when an end user or a device on the Internet makes an HTTP(S) request to CloudFront, and the request arrives at the edge location closest to that user.
  - **Viewer Response** – event occurs when the CloudFront server at the edge is ready to respond to the end user or the device that made the request
  - **Origin Request** – event occurs when the CloudFront edge server does not already have the requested object cached, and the viewer request is ready to be sent to backend origin webserver
  - **Origin Response** – event occurs when the CloudFront server at the edge receives a response from your backend origin webserver.
- Lambda function executes across AWS locations globally when a request for content is received, and scales with the volume of CloudFront requests globally
- **Lambda@Edge only supports Node.js for global invocation by CloudFront events at this time**

Lambda Key notes

- Serverless, you don’t need to have the infrastructure to run a service
- Your grant AWS lambda permission to access a dynamodb stream using an IAM role known as the **“execution role”**

Lambda VS Beanstalk

Lambda is considered a Function as a Service, whereby you create a piece of code and AWS will execute that piece of code, and only that piece of code as many times as you tell it to.
An example of a Lambda function use case might be that you want AWS to run a simple script for you. If you did not use Lambda, you would have to build an EC2 instance, install an operating system and install application runtimes and then run your script. Taking this concept further, if you wanted to run that script thousands of times per second you would have to build multiple EC2 instances and scale them out in order to provide enough compute.
With Lambda, AWS scales out the underlying infrastructure that runs your code automatically. All you need to be concerned about is your code!
AWS Lambda is perfect for building applications that have been broken down into services aka micro-services.
Elastic Beanstalk is different and yes it is considered Platform as a Service. It is an orchestration service that automatically builds your EC2, autoscaling groups, ELB's, Cloudwatch metrics and S3 buckets so that you can focus on just deploying applications to AWS and not worry about infrastructure tasks! If you didn't use Elastic Beanstalk then it would take significantly longer and would require intimate knowledge of AWS in order to provide the same functionality for your developers
In summary, AWS Lambda runs your code, Elastic Beanstalk runs your applications

Lambda Key points

- **Examples of How to Use AWS Lambda.** The use cases for AWS Lambda can be grouped into the following categories:
    - **Using AWS Lambda with AWS services as event sources** – *Event sources* publish events that cause the Lambda function to be invoked. These can be AWS services such as Amazon S3. For more information and tutorials, see the following topics:
        - Using AWS Lambda with Amazon S3
        - Using AWS Lambda with Kinesis
        - Using AWS Lambda with Amazon SQS
        - Using AWS Lambda with Amazon DynamoDB
        - Using AWS Lambda with AWS CloudTrail

- Using AWS Lambda with Amazon SNS from Different Accounts

- **On-demand Lambda function invocation over HTTPS** (Amazon API Gateway) – In addition to invoking Lambda functions using event sources, you can also invoke your Lambda function over HTTPS. You can do this by defining a custom REST API and endpoint using API Gateway. For more information and a tutorial, see Using AWS Lambda with Amazon API Gateway (On-Demand Over HTTPS).

- **On-demand Lambda function invocation (build your own event sources using custom apps)** – User applications such as client, mobile, or web applications can publish events and invoke Lambda functions using the AWS SDKs or AWS Mobile SDKs, such as the AWS Mobile SDK for Android. For more information and a tutorial, see Getting Started and Using AWS Lambda as Mobile Application Backend (Custom Event Source: Android)

- **Scheduled events** – You can also set up AWS Lambda to invoke your code on a regular, scheduled basis using the AWS Lambda console. You can specify a fixed rate (number of hours, days, or weeks) or you can specify a cron expression. For more information and a tutorial, see Using AWS Lambda with Scheduled Events.

Supported Event Sources

This topic lists the supported AWS services that you can configure as event sources for AWS Lambda functions. After you preconfigure the event source mapping, your Lambda function gets invoked automatically when these event sources detect events. For more information about invocation modes, see Event Source Mapping.

- **Amazon S3**
  - You can write Lambda functions to process S3 bucket events, such as the object-created or object-deleted events. For example, when a user uploads a photo to a bucket, you might want Amazon S3 to invoke your Lambda function so that it reads the image and creates a thumbnail for the photo.
- **Amazon DynamoDB**
  - You can use Lambda functions as triggers for your Amazon DynamoDB table. Triggers are custom actions you take in response to updates made to the DynamoDB table. To create a trigger, first you enable Amazon DynamoDB Streams for your table. AWS Lambda polls the stream and your Lambda function processes any updates published to the stream (based on an event source).

- **Amazon Kinesis Data Streams**
  - You can configure AWS Lambda to automatically poll your stream and process any new records such as website click streams, financial transactions, social media feeds, IT logs, and location-tracking events. Then, AWS Lambda polls the stream periodically (once per second) for new records.
- **Amazon Simple Notification Service**
  - You can write Lambda functions to process Amazon Simple Notification Service notifications. When a message is published to an Amazon SNS topic, the service can invoke your Lambda function by passing the message payload as parameter. Your Lambda function code can then process the event, for example publish the message to other Amazon SNS topics, or send the message to other AWS services.
  - This also enables you to trigger a Lambda function in response to Amazon CloudWatch alarms and other AWS services that use Amazon SNS.
- **Amazon Simple Email Service**
  - Amazon Simple Email Service (Amazon SES) is a cost-effective email service. With Amazon SES, in addition to sending emails, you can also use the service to receive messages. When you use Amazon SES to receive messages, you can configure Amazon SES to call your Lambda function when messages arrive. The service can then invoke your Lambda function by passing in the incoming email event, which in reality is an Amazon SES message in an Amazon SNS event, as a parameter. For example scenarios, see Considering Your Use Case for Amazon SES Email Receiving.
- **Amazon Simple Queue Service**
  - Amazon Simple Queue Service (Amazon SQS) allows you to build asynchronous workflows. You can configure AWS Lambda to poll for these messages as they arrive and then pass the event to a Lambda function invocation. To view a sample event, see Amazon SQS Event.
  - To set up Amazon Simple Queue Service as an event source for AWS Lambda, you first create or update an Amazon SQS queue and select custom values for the queue parameters. The following parameters will impact Amazon SQS's polling behavior:
  - **VisibilityTimeout**: May impact the period between retries.
  - **TimeToWait**: Will determine long poll duration. The default value is 20 seconds.
- **Amazon Cognito**

- The Amazon Cognito Events feature enables you to run Lambda function in response to events in Amazon Cognito. For example, you can invoke a Lambda function for the Sync Trigger events, that is published each time a dataset is synchronized.

- **AWS CloudFormation**
  - As part of deploying AWS CloudFormation stacks, you can specify a Lambda function as a custom resource to execute any custom commands. Associating a Lambda function with a custom resource enables you to invoke your Lambda function whenever you create, update, or delete AWS CloudFormation stacks.
- **Amazon CloudWatch Logs**
  - You can use AWS Lambda functions to perform custom analysis on Amazon CloudWatch Logs using CloudWatch Logs subscriptions. CloudWatch Logs subscriptions provide access to a real-time feed of log events from CloudWatch Logs and deliver it to your AWS Lambda function for custom processing, analysis, or loading to other systems. For more information about CloudWatch Logs, see Monitoring Log Files.
- **Amazon CloudWatch Events**
  - Amazon CloudWatch Events help you to respond to state changes in your AWS resources. When your resources change state, they automatically send events into an event stream. You can create rules that match selected events in the stream and route them to your AWS Lambda function to take action. For example, you can automatically invoke an AWS Lambda function to log the state of an EC2 instance or AutoScaling Group.
- **AWS CodeCommit**
  - You can create a trigger for an AWS CodeCommit repository so that events in the repository will invoke a Lambda function. For example, you can invoke a Lambda function when a branch or tag is created or when a push is made to an existing branch. For more information, see Manage Triggers for an AWS CodeCommit Repository.
- **Scheduled Events (powered by Amazon CloudWatch Events)**
  - You can also set up AWS Lambda to invoke your code on a regular, scheduled basis using the schedule event capability in Amazon CloudWatch Events. To set a schedule you can specify a fixed rate (number of hours, days, or weeks) or specify a cron expression (see Schedule Expression Syntax for Rules in the *Amazon CloudWatch User Guide*).
- **AWS Config**
  - You can use AWS Lambda functions to evaluate whether your AWS resource configurations comply with your custom Config rules. As

resources are created, deleted, or changed, AWS Config records these changes and sends the information to your Lambda functions. Your Lambda functions then evaluate the changes and report results to AWS Config. You can then use AWS Config to assess overall resource compliance: you can learn which resources are noncompliant and which configuration attributes are the cause of noncompliance.

- **Amazon Alexa**
  - You can use Lambda functions to build services that give new skills to Alexa, the Voice assistant on Amazon Echo. The Alexa Skills Kit provides the APIs, tools, and documentation to create these new skills, powered by your own services running as Lambda functions. Amazon Echo users can access these new skills by asking Alexa questions or making requests. For more information, see:
- **Amazon Lex**
  - Amazon Lex is an AWS service for building conversational interfaces into applications using voice and text. Amazon Lex provides pre-build integration with AWS Lambda, allowing you to create Lambda functions for use as code hook with your Amazon Lex bot. In your intent configuration, you can identify your Lambda function to perform initialization/validation, fulfillment, or both.
- **Amazon API Gateway**
  - You can invoke a Lambda function over HTTPS. You can do this by defining a custom REST API endpoint using Amazon API Gateway. You map individual API operations, such as `GET` and `PUT`, to specific Lambda functions. When you send an HTTPS request to the API endpoint, the Amazon API Gateway service invokes the corresponding Lambda function.
  - In addition, you can also use Lambda functions with other AWS services that publish data to one of the supported AWS event sources listed in this topic. For example, you can:
    - **Trigger Lambda functions in response to CloudTrail updates** because it records all API access events to an Amazon S3 bucket.
    - **Trigger Lambda functions in response to CloudWatch alarms** because it publishes alarm events to an Amazon SNS topic.
- **AWS IoT Button**
  - The AWS IoT button is a programmable button based on the Amazon Dash Button hardware. This simple Wi-Fi device is easy to configure and designed for developers to get started with AWS Lambda, among many other AWS services, without writing device-specific code.
- **Amazon CloudFront**

- Lambda@Edge lets you run Lambda functions at AWS Regions and Amazon CloudFront edge locations in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:
  - After CloudFront receives a request from a viewer (viewer request)
  - Before CloudFront forwards the request to the origin (origin request)
  - After CloudFront receives the response from the origin (origin response)
  - Before CloudFront forwards the response to the viewer (viewer response)

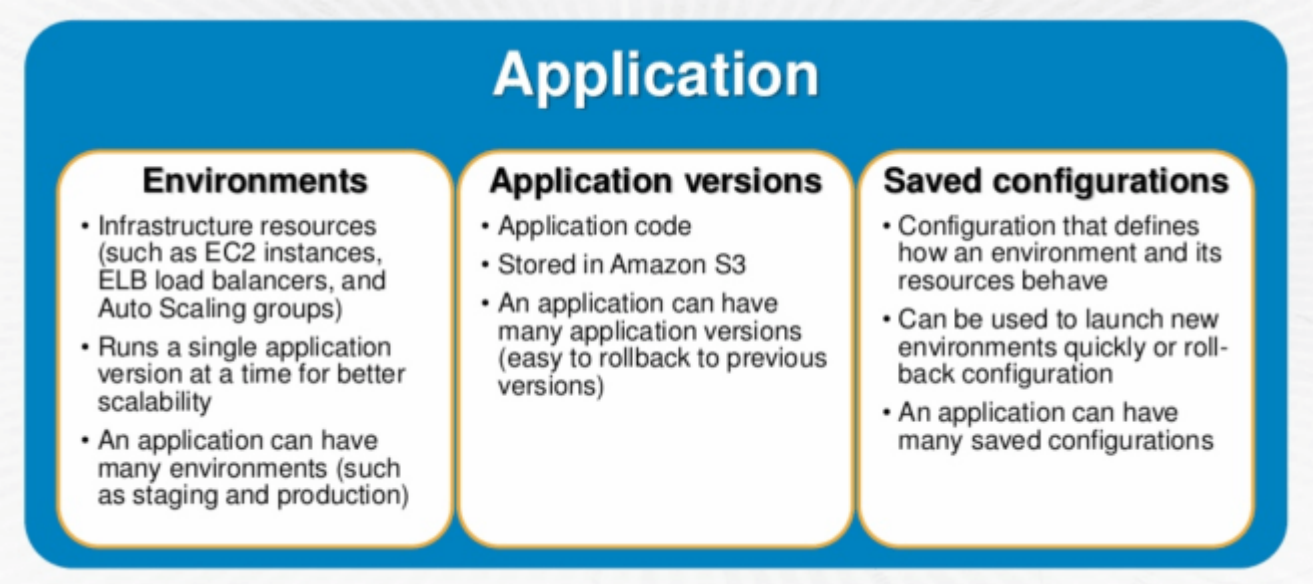
- **Amazon Kinesis Data Firehose**
  - Amazon Kinesis Data Firehose is the easiest way to load streaming data into AWS. It can capture, transform, and load streaming data into downstream services such as Kinesis Data Analytics or Amazon S3, enabling near real-time analytics with existing business intelligence tools and dashboards you're already using today. You can write Lambda functions to request additional, customized processing of the data before it is sent downstream.

- **Other Event Sources: Invoking a Lambda Function On Demand**
  - In addition to invoking Lambda functions using event sources, you can also invoke your Lambda function on demand. You don't need to preconfigure any event source mapping in this case. However, make sure that the custom application has the necessary permissions to invoke your Lambda function.
    - For example, user applications can also generate events (build your own custom event sources). User applications such as client, mobile, or web applications can publish events and invoke Lambda functions using the AWS SDKs or AWS Mobile SDKs such as the AWS Mobile SDK for Android.
    - For more information, see Tools for Amazon Web Services. For an example tutorial, see Using AWS Lambda with Amazon API Gateway (On-Demand Over HTTPS).

AWS Elastic Beanstalk

- **AWS Elastic Beanstalk helps to quickly deploy and manage applications in the AWS Cloud without having to worry about the infrastructure that runs those applications.**
- To deploy and scale web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.
- Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.
- There is no additional charge for Elastic Beanstalk - you pay only for the **AWS resources** needed to store and run your applications.
- Elastic Beanstalk reduces management complexity without restricting choice or control.
- Elastic Beanstalk enables automated infrastructure management and code deployment, by simply uploading, for applications and includes
    - **Application platform management**
    - **Capacity provisioning**
    - **Load Balancing**
    - **Auto scaling**
    - **Code deployment**
    - **Health Monitoring**
- Once an application is uploaded, Elastic Beanstalk automatically launches an environment, creates and configures the AWS resources needed to run the code. After your environment is launched, it can be managed and used to deploy new application versions
- AWS resources launched by Elastic Beanstalk are fully accessible i.e. EC2 instances can be accessed into
- Elastic Beanstalk provides developers and systems administrators an easy, fast way to deploy and manage the applications without having to worry about AWS infrastructure.
- CloudFormation, using templates, is a better option if the internal AWS resources to be used are known and fine grained control is needed

Elastic Beanstalk Components



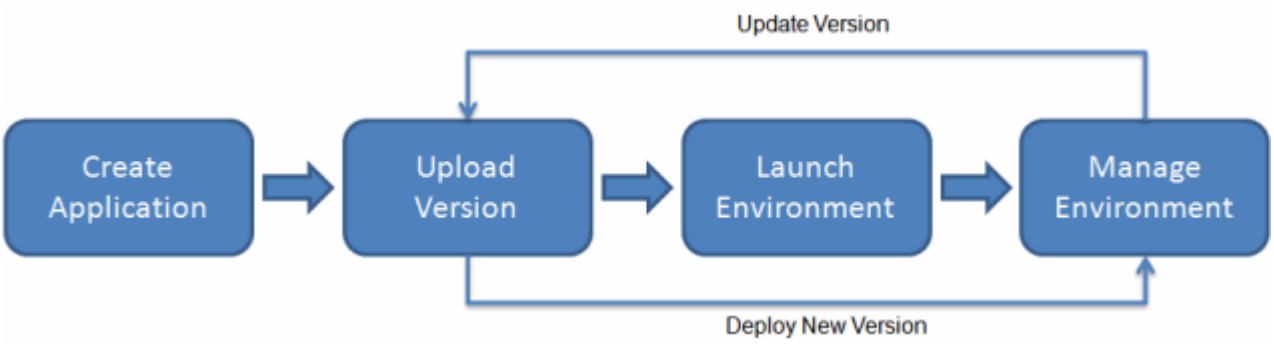
- **Application**
  - An Elastic Beanstalk application is a logical collection of Elastic Beanstalk components, including environments, versions, and environment configurations.

- **Application Version**
  - An application version refers to a specific, labelled iteration of deployable code for a web application
  - Applications can have many versions and each application version is unique and points to an S3 object
  - Multiple versions can be deployed for an Application for testing differences and helps roll-back to any version if case of issues

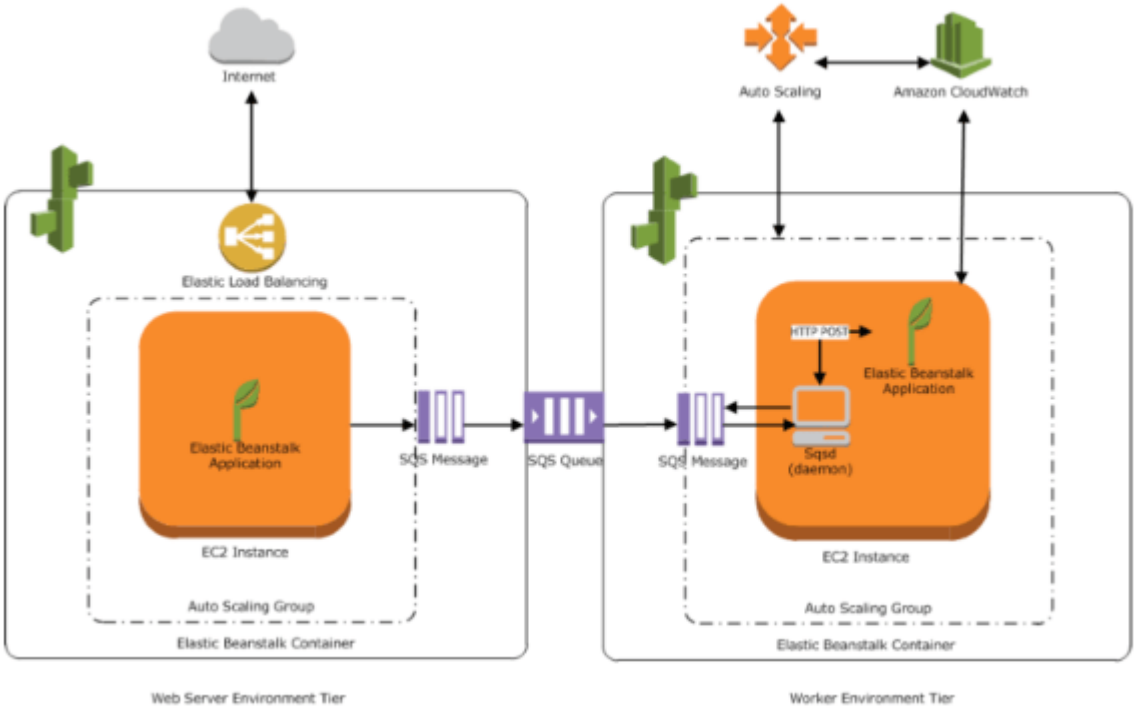
- **Environment**
  - An environment is a version that is deployed onto AWS resources
  - An environment runs a single application version at a time, but same application version can be deployed across multiple environments
  - When an environment is created, Elastic Beanstalk provisions the resources needed to run the application version you specified.

- **Environment Configuration**
  - An environment configuration identifies a collection of parameters and settings that define how an environment and its associated resources behave
  - When an environment's configuration settings is updated, Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources, depending upon the change

- **Configuration Template**
  - A configuration template is a starting point for creating unique environment configurations



- ## Elastic Beanstalk Architecture



- Elastic Beanstalk environment requires an environment tier, platform, and environment type
- Environment tier determines whether Elastic Beanstalk provisions resources to support a web application that handles HTTP(S) requests or a web application that handles background-processing tasks

- Web Environment Tier:
  - **An environment tier whose web application processes web requests is known as a web server tier.**
  - AWS resources created for a web environment tier include a Elastic Load Balancer, an Auto Scaling group, one or more EC2 instances

        - Every Environment has a CNAME url pointing to the ELB, aliased in Route 53 to ELB url
        - Each EC2 server instance that runs the application uses a container type, which defines the infrastructure topology and software stack
        - A software component called the host manager (HM) runs on each EC2 server instance and is responsible for
            - Deploying the application
            - Aggregating events and metrics for retrieval via the console, the API, or the command line
            - Generating instance-level events
            - Monitoring the application log files for critical errors
            - Monitoring the application server
            - Patching instance components
            - Rotating your application's log files and publishing them to S3

- Worker Environment Tier:
    - **An environment tier whose web application runs background jobs is known as a worker tier**
    - **AWS resources created for a worker environment tier include an Auto Scaling group, one or more Amazon EC2 instances, and an IAM role.**
    - For the worker environment tier, Elastic Beanstalk also creates and provisions an SQS queue, if one doesn't exist
    - When a worker environment tier is launched, Elastic Beanstalk installs the necessary support files for the programming language of choice and a daemon on each EC2 instance in the Auto Scaling group reading from the same SQS queue
    - Daemon is responsible for pulling requests from an SQS queue and then sending the data to the web application running in the worker environment tier that will process those messages
    - Elastic Beanstalk worker environments support SQS dead letter queues which can be used to store messages that could not be successfully processed. Dead letter queue provides the ability to sideline, isolate and analyze the unsuccessfully processed messages

- One environment cannot support two different environment tiers because each requires its own set of resources; a worker environment tier and a web server environment tier each require an Auto Scaling group, but Elastic Beanstalk supports only one Auto Scaling group per environment.

Elastic Beanstalk with other AWS Services

- **Elastic Beanstalk supports VPC and launches AWS resources, such as instances, into the VPC**
- **Elastic Beanstalk supports IAM and helps you securely control access to your AWS resources.**
- CloudFront can be used to distribute the content in S3, after an Elastic Beanstalk is created and deployed
- **CloudTrail**

- Elastic Beanstalk is integrated with CloudTrail, a service that captures all of the Elastic BeanstalkAPI calls and delivers the log files to an S3 bucket that you specify.
- CloudTrail captures API calls from the Elastic Beanstalk console or from your code to the Elastic Beanstalk APIs and help to determine the request made to Elastic Beanstalk, the source IP address from which the request was made, who made the request, when it was made etc.

- **RDS**
  - Elastic Beanstalk provides support for running RDS instances in the Elastic Beanstalk environment which is ideal for development and testing but not for production.
  - For a production environment, it is not recommended because it ties the lifecycle of the database instance to the lifecycle of application's environment. So it the Elastic beanstalk environment is deleted, the RDS instance is deleted as well
  - It is recommended to launch a database instance outside of the environment and configure the application to connect to it outside of the functionality provided by Elastic Beanstalk.
  - Using a database instance external to your environment requires additional security group and connection string configuration, but it also lets the application connect to the database from multiple environments, use database types not supported with integrated databases, perform blue/green deployments, and tear down your environment without affecting the database instance.

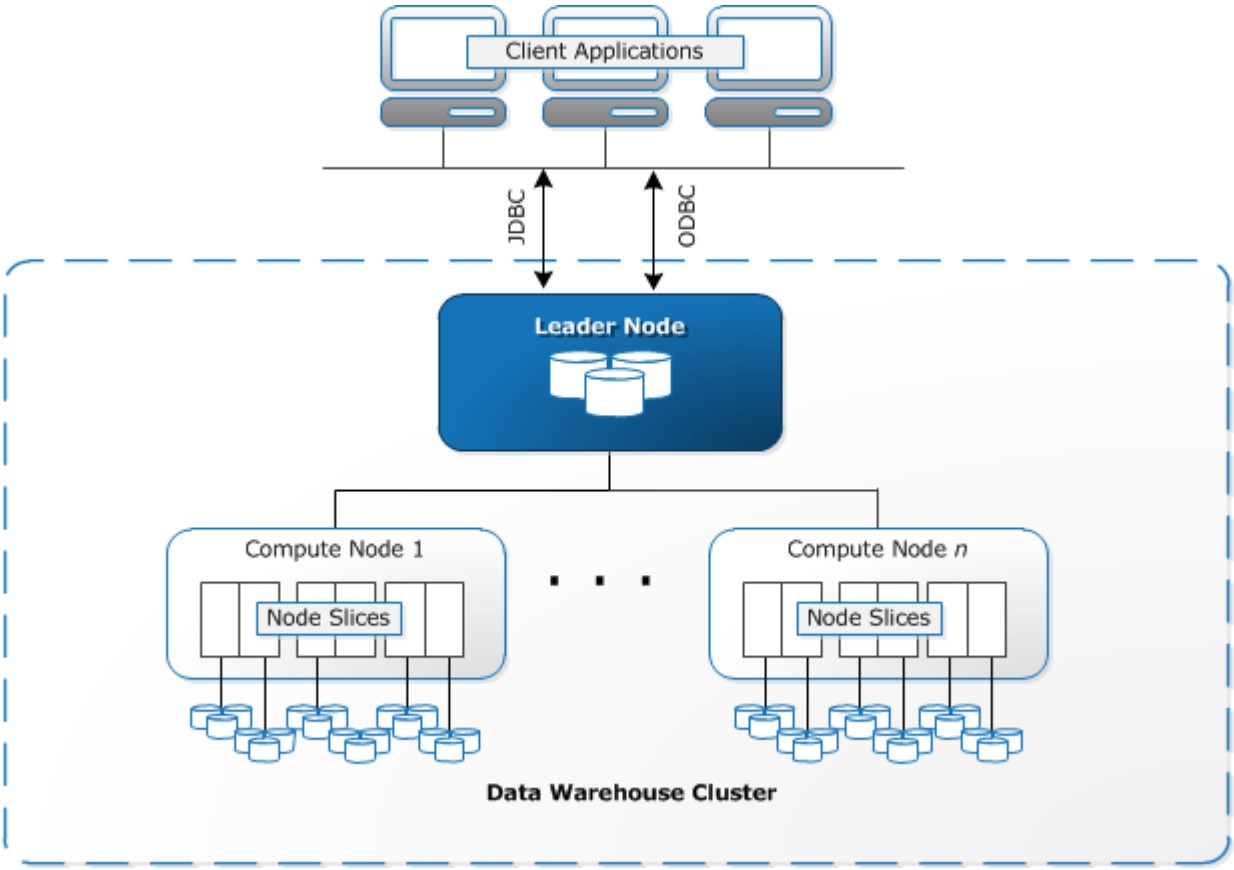
- **S3**
  - Elastic Beanstalk creates an S3 bucket named elasticbeanstalk-region-account-id for each region in which environments is created.
  - Elastic Beanstalk uses the bucket to store application versions, logs, and other supporting files.
  - It applies a bucket policy to buckets it creates to allow environments to write to the bucket and prevent accidental deletion

AWS Redshift

Amazon Redshift is a fast, fully managed that makes it simple and cost-effective to analyze all your data using standard SQL and your existing Business Intelligence (BI) tools. It allows you to run complex analytic queries against petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution. Most results come back in seconds.
Amazon Redshift also includes , allowing you to directly run SQL queries against exabytes of unstructured data in . No loading or transformation is required, and you can use open data formats, including Avro, CSV, Grok, Ion, JSON, ORC, Parquet, RCFile, RegexSerDe, SequenceFile, TextFile, and TSV. Redshift Spectrum automatically scales query compute capacity based on the data being retrieved, so queries against Amazon S3 run fast, regardless of data set

- **Redshift automatically helps**
    - set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity
    - patches and backs up the data warehouse, storing the backups for a user-defined retention period
    - monitors the nodes and drives to help recovery from failures
    - significantly lowers the cost of a data warehouse, but also makes it easy to analyze large amounts of data very quickly
    - provide fast querying capabilities over structured data using familiar SQL-based clients and business intelligence (BI) tools using standard **ODBC and JDBC connections.**
    - uses replication and continuous backups to enhance availability and improve data durability and can automatically recover from node and component failures.
    - scale up or down with a few clicks in the AWS Management Console or with a single API call
    - distribute & parallelize queries across multiple physical resources
    - **supports VPC, SSL, AES-256 encryption and Hardware Security Modules (HSMs) to protect the data in transit and at rest.**

- Redshift only supports Single-AZ deployments and the nodes are available within the same AZ, if the AZ supports Redshift clusters
- Redshift provides monitoring using CloudWatch and metrics for compute utilization, storage utilization, and read/write traffic to the cluster are available with the ability to add user-defined custom metrics
- Redshift provides Audit logging and AWS CloudTrail integration
- Redshift can be easily enabled to a second region for disaster recovery.



- ## Redshift Performance

- **Massively Parallel Processing (MPP)**
  - automatically distributes data and query load across all nodes.
  - makes it easy to add nodes to the data warehouse and enables fast query performance as the data warehouse grows.

- **Columnar Data Storage**
  - organizes the data by column, as column-based systems are ideal for data warehousing and analytics, where queries often involve aggregates performed over large data sets
  - columnar data is stored sequentially on the storage media, and require far fewer I/Os, greatly improving query performance

- **Advance Compression**
  - Columnar data stores can be compressed much more than row-based data stores because similar data is stored sequentially on disk.
  - employs multiple compression techniques and can often achieve significant compression relative to traditional relational data stores.
  - doesn’t require indexes or materialized views and so uses less space than traditional relational database systems.
  - automatically samples the data and selects the most appropriate compression scheme, when the data is loaded into an empty table

Redshift Single vs Multi-Node Cluster

- **Single Node**
  - single node configuration enables getting started quickly and cost-effectively & scale up to a multi-node configuration as the needs grow

- **Multi-Node**
  - Multi-node configuration requires a leader node that manages client connections and receives queries, and two or more compute nodes that store data and perform queries and computations.
  - Leader node
    - provisioned automatically and not charged for receives queries from client applications, parses the queries and develops execution plans, which are an ordered set of steps to process these queries.
    - coordinates the parallel execution of these plans with the compute nodes, aggregates the intermediate results from these nodes and finally returns the results back to the client applications.
  - **Compute node**
    - can contain from 1-128 compute nodes, depending on the node type
    - executes the steps specified in the execution plans and transmit data among themselves to serve these queries.
    - intermediate results are sent back to the leader node for aggregation before being sent back to the client applications.
    - Supports Dense Storage or Dense Compute nodes (DC) instance type
      - Dense Storage (DS) allow creation of very large data warehouses using hard disk drives (HDDs) for a very low price point
      - Dense Compute (DC) allow creation of very high performance data warehouses using fast CPUs, large amounts of RAM and solid-state disks (SSDs)
    - direct access to compute nodes is not allowed

Redshift Availability & Durability

- Redshift replicates the data within the data warehouse cluster and continuously backs up the data to S3 (11 9’s durability)
- Redshift mirrors each drive’s data to other nodes within the cluster.
- Redshift will automatically detect and replace a failed drive or node
- If a drive fails, Redshift
  - cluster will remain available in the event of a drive failure
  - the queries will continue with a slight latency increase while Redshift rebuilds the drive from replica of the data on that drive which is stored on other drives within that node
  - single node clusters do not support data replication and the cluster needs to be restored from snapshot on S3

- **In case of node failure(s), Redshift**
    - **automatically provisions new node(s) and begins restoring data from other drives within the cluster or from S3**
    - prioritizes restoring the most frequently queried data so the most frequently executed queries will become per-formant quickly
    - cluster will be unavailable for queries and updates until a replacement node is provisioned and added to the cluster

- **In case of Redshift cluster AZ goes down, Redshift**
    - **cluster is unavailable until power and network access to the AZ are restored**
    - cluster's data is preserved and can be used once AZ becomes available
    - cluster can be restored from any existing snapshots to a new AZ within the same region

Redshift Backup & Restore

- Redshift replicates all the data within the data warehouse cluster when it is loaded and also continuously backs up the data to S3
- Redshift always attempts to maintain at least three copies of the data
- Redshift enables automated backups of the data warehouse cluster with a 1-day retention period, by default, which can be extended to max 35 days
- Automated backups can be turned off by setting the retention period as 0
- Redshift can also asynchronously replicate the snapshots to S3 in another region for disaster recovery

Redshift Scalability

- Redshift allows scaling of the cluster either by
    - increasing the node instance type (Vertical scaling)
    - increasing the number of nodes (Horizontal scaling)
- Redshift scaling changes are usually applied during the maintenance window or can be applied immediately
- Redshift scaling process
    - existing cluster remains available for read operations only while a new data warehouse cluster gets created during scaling operations
    - data from the compute nodes in the existing data warehouse cluster is moved in parallel to the compute nodes in the new cluster
    - when the new data warehouse cluster is ready, the existing cluster will be temporarily unavailable while the canonical name record of the existing cluster is flipped to point to the new data warehouse cluster

Redshift vs EMR vs RDS

- **RDS is ideal for**
  - structured data and running traditional relational databases while offloading database administration
  - for online-transaction processing (OLTP) and for reporting and analysis

- **Redshift is ideal for**
  - large volumes of structured data that needs to be persisted and queried using standard SQL and existing BI tools
  - analytic and reporting workloads against very large data sets by harnessing the scale and resources of multiple nodes and using a variety of optimizations to provide improvements over RDS
  - preventing reporting and analytic processing from interfering with the performance of the OLTP workload

- **EMR is ideal for**
  - processing and transforming unstructured or semi-structured data to bring in to Amazon Redshift and
  - for data sets that are relatively transitory, not stored for long-term use.

Amazon Redshift Database Encryption

- **In Amazon Redshift, you can enable database encryption for your clusters to help protect data at rest. When you enable encryption for a cluster, the data blocks and system metadata are encrypted for the cluster and its snapshots.**

- Encryption is an optional, immutable setting of a cluster. If you want encryption, you enable it during the cluster launch process. To go from an unencrypted cluster to an encrypted cluster or the other way around, unload your data from the existing cluster and reload it in a new cluster with the chosen encryption setting.

- **Though encryption is an optional setting in Amazon Redshift, we recommend enabling it for clusters that contain sensitive data.** Additionally, you might be required to use encryption depending on the guidelines or regulations that govern your data. For example, the Payment Card Industry Data Security Standard (PCI DSS), the Sarbanes-Oxley Act (SOX), the Health Insurance Portability and Accountability Act (HIPAA), and other such regulations provide guidelines for handling specific types of data.
-
- Amazon Redshift uses a hierarchy of encryption keys to encrypt the database. **You can use either AWS Key Management Service (AWS KMS) or a hardware security module (HSM) to manage the top-level encryption keys in this hierarchy.** The process that Amazon Redshift uses for encryption differs depending on how you manage keys.

- **Additionally, Amazon Redshift automatically integrates with AWS KMS but not with an HSM. When you use an HSM, you must use client and server certificates to configure a trusted connection between Amazon Redshift and your HSM.**

Database Encryption for Redshift - Using AWS KMS

- **When you choose AWS KMS for key management with Amazon Redshift, there is a four-tier hierarchy of encryption keys. These keys, in hierarchical order, are the master key, a cluster encryption key (CEK), a database encryption key (DEK), and data encryption keys.**

- When you launch your cluster, Amazon Redshift returns a list of the customer master keys (CMKs) that your AWS account has created or has permission to use in AWS KMS. You select a CMK to use as your master key in the encryption hierarchy.

- **By default, Amazon Redshift selects your default key as the master key (CMK).** Your default key is an AWS-managed key that is created for your AWS account to use in Amazon Redshift. **AWS KMS creates this key the first time you launch an encrypted cluster in a region and choose the default key.**

- If you don’t want to use the default key, you must have (or create) a customer-managed CMK separately in AWS KMS before you launch your cluster in Amazon Redshift. Customer-managed CMKs give you more flexibility, including the ability to create, rotate, disable, define access control for, and audit the encryption keys used to help protect your data
.
- **If you want to use a AWS KMS key from another AWS account, you must have permission to use the key and specify its ARN in Amazon Redshift.**

- **After you choose a master key, Amazon Redshift requests that AWS KMS generate a data key and encrypt it using the selected master key. This data key is used as the CEK** in Amazon Redshift. AWS KMS exports the encrypted CEK to Amazon Redshift, where it is stored internally on disk in a separate network from the cluster along with the grant to the CMK and the encryption context for the CEK. Only the encrypted CEK is exported to Amazon Redshift; the CMK remains in AWS KMS. Amazon Redshift also passes the encrypted CEK over a secure channel to the cluster and loads it into memory. Then, Amazon Redshift calls AWS KMS to decrypt the CEK and loads the decrypted CEK into memory.
- **Next, Amazon Redshift randomly generates a key to use as the DEK and loads it into memory in the cluster. The decrypted CEK is used to encrypt the DEK,** which is then passed over a secure channel

from the cluster to be stored internally by Amazon Redshift on disk in a separate network from the cluster. Like the CEK, both the encrypted and decrypted versions of the DEK are loaded into memory in the cluster. The decrypted version of the DEK is then used to encrypt the individual encryption keys that are randomly generated for each data block in the database.

- When the cluster reboots, Amazon Redshift starts with the internally stored, encrypted versions of the CEK and DEK, reloads them into memory, and then calls AWS KMS to decrypt the CEK with the CMK again so it can be loaded into memory. The decrypted CEK is then used to decrypt the DEK again, and the decrypted DEK is loaded into memory and used to encrypt and decrypt the data block keys as needed.

- **Copying AWS KMS-Encrypted Snapshots to Another Region**
  - **AWS KMS keys are specific to a region.** If you enable copying of Amazon Redshift snapshots to another region, and the source cluster and its snapshots are encrypted using a master key from AWS KMS, you need to configure a grant for Amazon Redshift to use a master key in the destination region. This grant enables Amazon Redshift to encrypt snapshots in the destination region.

  - **Note:**
  - If you enable copying of snapshots from an encrypted cluster and use AWS KMS for your master key, you cannot rename your cluster because the cluster name is part of the encryption context. If you must rename your cluster, you can disable copying of snapshots in the source region, rename the cluster, and then configure and enable copying of snapshots again.

  - The process to configure the grant for copying snapshots is as follows.
    - In the destination region, create a snapshot copy grant by doing the following:
      - If you do not already have an AWS KMS key to use, create one.
      - Specify a name for the snapshot copy grant. This name must be unique in that region for your AWS account.
      - Specify the AWS KMS key ID for which you are creating the grant. If you do not specify a key ID, the grant applies to your default key.
    - In the source region, enable copying of snapshots and specify the name of the snapshot copy grant that you created in the destination region.

  - This preceding process is only necessary if you enable copying of snapshots using the AWS CLI, the Amazon Redshift API, or SDKs. **If you use the console, Amazon Redshift provides the proper workflow to configure the grant when you enable cross-region snapshot copy**.
  - Before the snapshot is copied to the destination region, Amazon Redshift decrypts the snapshot using the master key in the source region and re-encrypts it temporarily using a randomly generated RSA

key that Amazon Redshift manages internally. Amazon Redshift then copies the snapshot over a secure channel to the destination region, decrypts the snapshot using the internally managed RSA key, and then re-encrypts the snapshot using the master key in the destination region.

Encryption for Amazon Redshift Using Hardware Security Modules

- **If you don’t use AWS KMS for key management, you can use a hardware security module (HSM) for key management with Amazon Redshift.**
- **Important: HSM encryption is not supported for DC2 node types.**
- **HSMs are devices that provide direct control of key generation and management. They provide greater security by separating key management from the application and database layers.**
- Amazon Redshift supports AWS CloudHSM Classic for key management. The encryption process is different when you use HSM to manage your encryption keys instead of AWS KMS.
- **Important: Amazon Redshift supports only AWS CloudHSM Classic.** We don't support the newer AWS CloudHSM service. AWS CloudHSM Classic isn't available in all regions.
- **When you configure your cluster to use an HSM, Amazon Redshift sends a request to the HSM to generate and store a key to be used as the CEK.** However, unlike AWS KMS, the HSM doesn’t export the CEK to Amazon Redshift. Instead, **Amazon Redshift randomly generates the DEK in the cluster and passes it to the HSM to be encrypted by the CEK. The HSM returns the encrypted DEK to Amazon Redshift, where it is further encrypted using a randomly-generated, internal master key and stored internally on disk in a separate network from the cluster. Amazon Redshift also loads the decrypted version of the DEK in memory in the cluster so that the DEK can be used to encrypt and decrypt the individual keys for the data blocks.**
- If the cluster is rebooted, Amazon Redshift decrypts the internally-stored, double-encrypted DEK using the internal master key to return the internally stored DEK to the CEK-encrypted state. The CEK-encrypted DEK is then passed to the HSM to be decrypted and passed back to Amazon Redshift, where it can be loaded in memory again for use with the individual data block keys.
- **Configuring a Trusted Connection Between Amazon Redshift and an HSM**

- **When you opt to use an HSM for management of your cluster key, you need to configure a trusted network link between Amazon Redshift and your HSM. Doing this requires configuration of client and server certificates. The trusted connection is used to pass the encryption keys between the HSM and Amazon Redshift during encryption and decryption operations.**

- Amazon Redshift creates a public client certificate from a randomly generated private and public key pair. These are encrypted and stored internally. You download and register the public client certificate in your HSM, and assign it to the applicable HSM partition.

- You provide Amazon Redshift with the HSM IP address, HSM partition name, HSM partition password, and a public HSM server certificate, which is encrypted by using an internal master key. Amazon Redshift completes the configuration process and verifies that it can connect to the HSM. If it cannot, the cluster is put into the INCOMPATIBLE_HSM state and the cluster is not created. In this case, you must delete the incomplete cluster and try again.

- **Important: When you modify your cluster to use a different HSM partition, Amazon Redshift verifies that it can connect to the new partition, but it does not verify that a valid encryption key exists. Before you use the new partition, you must replicate your keys to the new partition. If the cluster is restarted and Amazon Redshift cannot find a valid key, the restart fails.**

About Rotating Encryption Keys in Amazon Redshift

- **In Amazon Redshift, you can rotate encryption keys for encrypted clusters. When you start the key rotation process, Amazon Redshift rotates the CEK for the specified cluster and for any automated or manual snapshots of the cluster. Amazon Redshift also rotates the DEK for the specified cluster, but cannot rotate the DEK for the snapshots while they are stored internally in Amazon Simple Storage Service (Amazon S3) and encrypted using the existing DEK.**

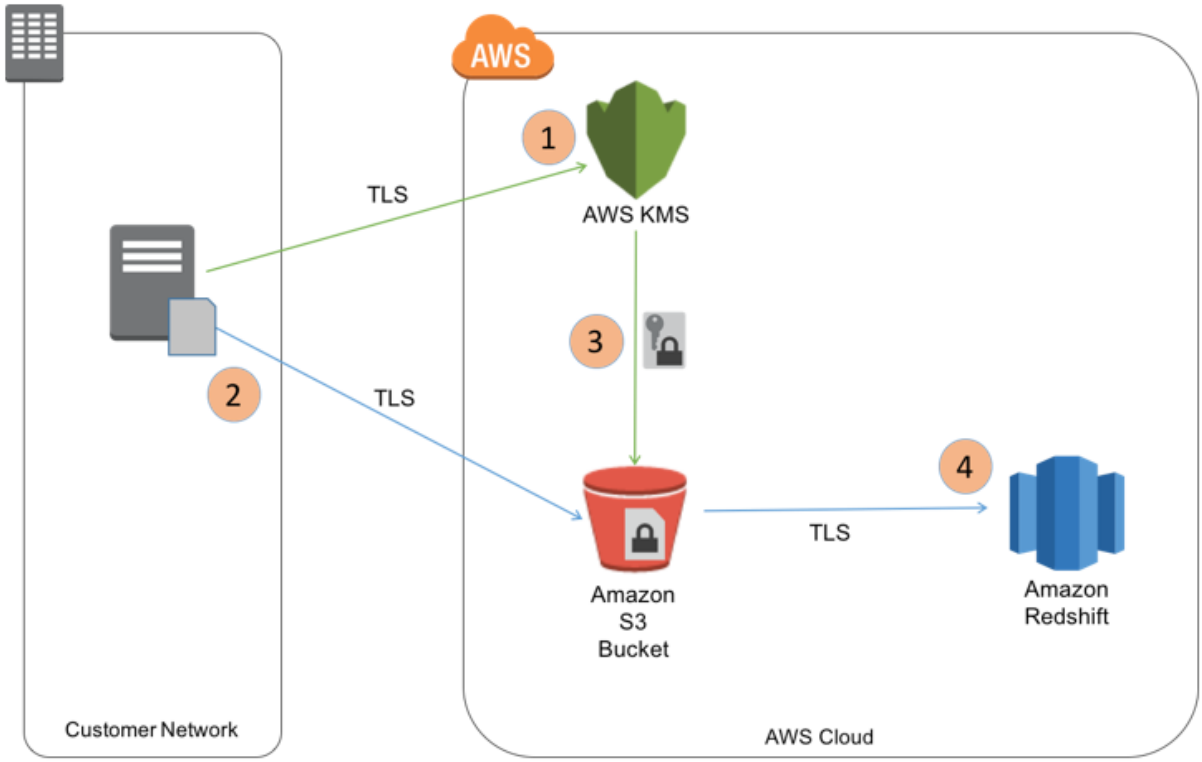
- **While the rotation is in progress, the cluster is put into a ROTATING_KEYS state until completion, at which time the cluster returns to the AVAILABLE state. Amazon Redshift handles decryption and re-encryption during the key rotation process.**

- **Note: You cannot rotate keys for snapshots without a source cluster. Before you delete a cluster, consider whether its snapshots rely on key rotation.**

- **Because the cluster is momentarily unavailable during the key rotation process, you should rotate keys only as often as your data needs require or when you suspect the keys might have been**

compromised. As a best practice, you should review the type of data that you store and plan how often to rotate the keys that encrypt that data. The frequency for rotating keys varies depending on your corporate policies for data security, and any industry standards regarding sensitive data and regulatory compliance. Ensure that your plan balances security needs with availability considerations for your cluster.

Encrypt Your Amazon Redshift Loads with Amazon S3 and AWS KMS



Redshift cluster data is not encrypted at rest
Redshift cluster data in the affected cluster is not encrypted at rest.

Details
In Amazon Redshift, you can enable database encryption for your clusters to help protect data at rest. When you enable encryption for a cluster, the data blocks and system metadata are encrypted for the cluster and its snapshots. Redshift cluster data in the affected cluster is not encrypted at rest.
Suggested Action
Encryption is an optional, immutable setting of a cluster. To encrypt the data in all your user-created tables, you can enable cluster encryption when you launch the cluster. If you want to go from an encrypted cluster to an unencrypted cluster or the other way around, you must unload your data from the existing cluster and reload it in a new cluster with the chosen encryption setting.

- **Amazon Redshift Enhanced VPC Routing**
  - When you use Amazon Redshift Enhanced VPC Routing, Amazon Redshift forces all and traffic between your cluster and your data

repositories through your Amazon VPC. You can now use standard VPC features to tightly manage the flow of data between your Amazon Redshift cluster and other resources. When you use Enhanced VPC Routing to route traffic through your VPC, you can also use to monitor COPY and UNLOAD traffic.

-
- **If Enhanced VPC Routing is not enabled, Amazon Redshift routes traffic through the Internet, including traffic to other services within the AWS network.**

For example, you can configure the following pathways in your VPC:

- **VPC Endpoints** – For traffic to an Amazon S3 bucket in the same region as your cluster, you can create a VPC endpoint to direct traffic directly to the bucket. When you use VPC endpoints, you can attach an endpoint policy to manage access to Amazon S3.

- **NAT gateway** – To connect to an Amazon S3 bucket in another region or to another service within the AWS network, or to access a host instance outside the AWS network

- **Internet gateway** – To connect to AWS services outside your VPC, you can attach an to your VPC subnet.

RedShift Key notes

- Amazon Redshift is a fast, fully managed that makes it simple and cost-effective to analyze all your data using standard SQL and your existing Business Intelligence (BI) tools. It allows you to run complex analytic queries against petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance local disks, and massively parallel query execution. Most results come back in seconds.

- You can easily scale an an Amazon Redshift data warehouse up or down with a few clicks in the AWS Management Console or with a single API call.

- **Can store huge amount of data** (a database) **but cannot ingest huge amounts of data in real time** (not like Kinesis can do)

- If you delete the cluster:
  - You can choose to have a final snapshot to use later
  - Manual backups are not deleted automatically, you don not manually delete them, you will be charged standard S3 storage rates

- You can also add additional, user-defined metrics via AWS CloudWatch custom metric functionality

- Billing: Compute node hours, Backup Storage and Data transfer (just out the AWS or the AWS region)

- **Supports encryption of data “at rest”** using hardware accelerated AES-256 bits
  - By Default, AWS Redshift takes care of encryption key management
  - You can choose to manage your own keys through HSM (Hardware Security Modules) or AWS KMS (Key Management Service)

- **Supports SSL Encryption in-transit** between client applications and Redshift data warehouse cluster

- **Redshift automatically helps**
  - set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity
  - patches and backs up the data warehouse, storing the backups for a user-defined retention period
  - monitors the nodes and drives to help recovery from failures
  - significantly lowers the cost of a data warehouse, but also makes it easy to analyze large amounts of data very quickly
  - provide fast querying capabilities over structured data using familiar SQL-based clients and business intelligence (BI) tools using standard **ODBC and JDBC connections.**
  - uses replication and continuous backups to enhance availability and improve data durability and can automatically recover from node and component failures.
  - distribute & parallelize queries across multiple physical resources
  - **supports VPC, SSL, AES-256 encryption and Hardware Security Modules (HSMs) to protect the data in transit and at rest.**

- Redshift only supports Single-AZ deployments and the nodes are available within the same AZ, if the AZ supports Redshift clusters

- Redshift provides monitoring using CloudWatch and metrics for compute utilization, storage utilization, and read/write traffic to the cluster are available with the ability to add user-defined custom metrics

- Redshift provides Audit logging and AWS CloudTrail integration

- Redshift can be easily enabled to a second region for disaster recovery.
- **Redshift Performance**

  - **Massively Parallel Processing (MPP)**
    - automatically distributes data and query load across all nodes.

  - **Columnar Data Storage**
    - organizes the data by column, as column-based systems are ideal for data warehousing and analytics, where queries often involve aggregates performed over large data sets

  - **Advance Compression**

- Columnar data stores can be compressed much more than row-based data stores because similar data is stored sequentially on disk.

- **Redshift Availability & Durability**

    - Redshift will automatically detect and replace a failed drive or node
    -
        - **In case of node failure(s), Redshift automatically provisions new node(s) and begins restoring data from other drives within the cluster or from S3**
    -
        - **In case of Redshift cluster AZ goes down, Redshift cluster is unavailable until power and network access to the AZ are restored**

- **Redshift vs EMR vs RDS**

- **RDS is ideal for**
    - structured data and running traditional relational databases while offloading database administration
    - for online-transaction processing (OLTP) and for reporting and analysis
    -
- **Redshift is ideal for**
    - large volumes of structured data that needs to be persisted and queried using standard SQL and existing BI tools
    - analytic and reporting workloads against very large data sets by harnessing the scale and resources of multiple nodes and using a variety of optimizations to provide improvements over RDS
    - preventing reporting and analytic processing from interfering with the performance of the OLTP workload

- **EMR is ideal for**
    - processing and transforming unstructured or semi-structured data to bring in to Amazon Redshift and
    - for data sets that are relatively transitory, not stored for long-term use.
    -
- **Amazon Redshift Database Encryption**

    - In Amazon Redshift, you can enable database encryption for your clusters to help protect data at rest. When you enable encryption for a cluster, the data blocks and system metadata are encrypted for the cluster and its snapshots.

    - To go from an unencrypted cluster to an encrypted cluster or the other way around, unload your data from the existing cluster and reload it in a new cluster with the chosen encryption setting.

- Though encryption is an optional setting in Amazon Redshift, we recommend enabling it for clusters that contain sensitive data.

- Amazon Redshift uses a hierarchy of encryption keys to encrypt the database. **You can use either AWS Key Management Service (AWS KMS) or a hardware security module (HSM) to manage the top-level encryption keys in this hierarchy.** The process that Amazon Redshift uses for encryption differs depending on how you manage keys.

- Additionally, Amazon Redshift automatically integrates with AWS KMS but not with an HSM. When you use an HSM, you must use client and server certificates to configure a trusted connection between Amazon Redshift and your HSM.

- **Database Encryption for Amazon Redshift Using AWS KMS**

  - When you choose AWS KMS for key management with Amazon Redshift, there is a four-tier hierarchy of encryption keys. These keys, in hierarchical order, are the master key, a cluster encryption key (CEK), a database encryption key (DEK), and data encryption keys.

  - **By default, Amazon Redshift selects your default key as the master key (CMK).** Your default key is an AWS-managed key that is created for your AWS account to use in Amazon Redshift. **AWS KMS creates this key the first time you launch an encrypted cluster in a region and choose the default key.**

  - If you want to use a AWS KMS key from another AWS account, you must have permission to use the key and specify its ARN in Amazon Redshift.

  - After you choose a master key, Amazon Redshift requests that AWS:

    - KMS generate a data key and encrypt it using the selected master key. This data key is used as the CEK
    - Next, Amazon Redshift randomly generates a key to use as the DEK and loads it into memory in the cluster. The decrypted CEK is used to encrypt the DEK
    -

- **Copying AWS KMS-Encrypted Snapshots to Another Region**

  - **AWS KMS keys are specific to a region.** If you enable copying of Amazon Redshift snapshots to another region, and the source cluster and its snapshots are encrypted using a master key from AWS KMS, you need to configure a grant for Amazon Redshift to use a master key in the destination region. This grant enables Amazon Redshift to encrypt snapshots in the destination region.

- **If you use the console, Amazon Redshift provides the proper workflow to configure the grant when you enable cross-region snapshot copy**.

- **Encryption for Amazon Redshift Using Hardware Security Modules**

  - If you don’t use AWS KMS for key management, you can use a hardware security module (HSM) for key management with Amazon Redshift.

  - Important: HSM encryption is not supported for DC2 node types.

  - HSMs are devices that provide direct control of key generation and management. They provide greater security by separating key management from the application and database layers.

  - Important: Amazon Redshift supports only AWS CloudHSM Classic.

    - **When you configure your cluster to use an HSM, Amazon Redshift sends a request to the HSM to generate and store a key to be used as the CEK.** However, unlike AWS KMS, the HSM doesn’t export the CEK to Amazon Redshift. Instead, **Amazon Redshift randomly generates the DEK in the cluster and passes it to the HSM to be encrypted by the CEK.**

    - **The HSM returns the encrypted DEK to Amazon Redshift, where it is further encrypted using a randomly-generated, internal master key and stored internally on disk in a separate network from the cluster. Amazon Redshift also loads the decrypted version of the DEK in memory in the cluster so that the DEK can be used to encrypt and decrypt the individual keys for the data blocks.**
- **Configuring a Trusted Connection Between Amazon Redshift and an HSM**

- When you opt to use an HSM for management of your cluster key, you need to configure a trusted network link between Amazon Redshift and your HSM. Doing this requires configuration of client and server certificates. The trusted connection is used to pass the encryption keys between the HSM and Amazon Redshift during encryption and decryption operations.
  - When you modify your cluster to use a different HSM partition, Amazon Redshift verifies that it can connect to the new partition, but it does not verify that a valid encryption key exists. Before you use the new partition, you must replicate your keys to the new partition. **If the cluster is restarted and Amazon Redshift cannot find a valid key, the restart fails.**

Note: AWS CloudHSM provides hardware security modules in the AWS Cloud. A hardware security module (HSM) is a computing device that processes cryptographic operations and provides secure storage for cryptographic keys.

- **About Rotating Encryption Keys in Amazon Redshift**
    - In Amazon Redshift, you can rotate encryption keys for encrypted clusters. When you start the key rotation process, Amazon Redshift rotates the CEK for the specified cluster and for any automated or manual snapshots of the cluster. Amazon Redshift also rotates the DEK for the specified cluster, but cannot rotate the DEK for the snapshots while they are stored internally in Amazon Simple Storage Service (Amazon S3) and encrypted using the existing DEK.
    - While the rotation is in progress, the cluster is put into a ROTATING_KEYS state until completion, at which time the cluster returns to the AVAILABLE state. Amazon Redshift handles decryption and re-encryption during the key rotation process.
    - **Note: You cannot rotate keys for snapshots without a source cluster. Before you delete a cluster, consider whether its snapshots rely on key rotation.**
    - Because the cluster is momentarily unavailable during the key rotation process, you should rotate keys only as often as your data needs require or when you suspect the keys might have been compromised. As a best practice, you should review the type of data that you store and plan how often to rotate the keys that encrypt that data.
- **Amazon Redshift Enhanced VPC Routing**
    - When you use Amazon Redshift Enhanced VPC Routing, Amazon Redshift forces all COPYand traffic between your cluster and your data repositories through your Amazon VPC. You can now use standard VPC features to tightly manage the flow of data between your Amazon Redshift cluster and other resources.
    - When you use Enhanced VPC Routing to route traffic through your VPC, you can also use to monitor COPY and UNLOAD traffic.
    - **If Enhanced VPC Routing is not enabled, Amazon Redshift routes traffic through the Internet, including traffic to other services within the AWS network.**
        - For example, you can configure the following pathways in your VPC:
- **VPC Endpoints** – For traffic to an Amazon S3 bucket in the same region as your cluster, you can create a VPC endpoint to direct traffic directly to the bucket. When you use VPC endpoints, you can attach an endpoint policy to manage access to Amazon S3.
- **NAT gateway** – To connect to an Amazon S3 bucket in another region or to another service within the AWS network, or to access a host instance outside the AWS network

- **Internet gateway** – To connect to AWS services outside your VPC, you can attach an to your VPC subnet.

AWS Kinesis

- **Amazon Kinesis enables real-time processing of streaming data at massive scale**

- Kinesis Streams enables building of custom applications that process or analyze streaming data for specialized needs

- **Kinesis Streams features**
    - handles provisioning, deployment, ongoing-maintenance of hardware, software, or other services for the data streams
    - manages the infrastructure, storage, networking, and configuration needed to stream the data at the level of required data throughput
    - **synchronously replicates data across three facilities in an AWS Region, providing high availability and data durability**
    - stores records of a stream for up to 24 hours, by default, from the time they are added to the stream. The limit can be raised to up to 7 days by enabling extended data retention

- Data such as clickstreams, application logs, social media etc can be added from multiple sources and within seconds is available for processing to the Amazon Kinesis Applications

- Kinesis provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple Kinesis applications.

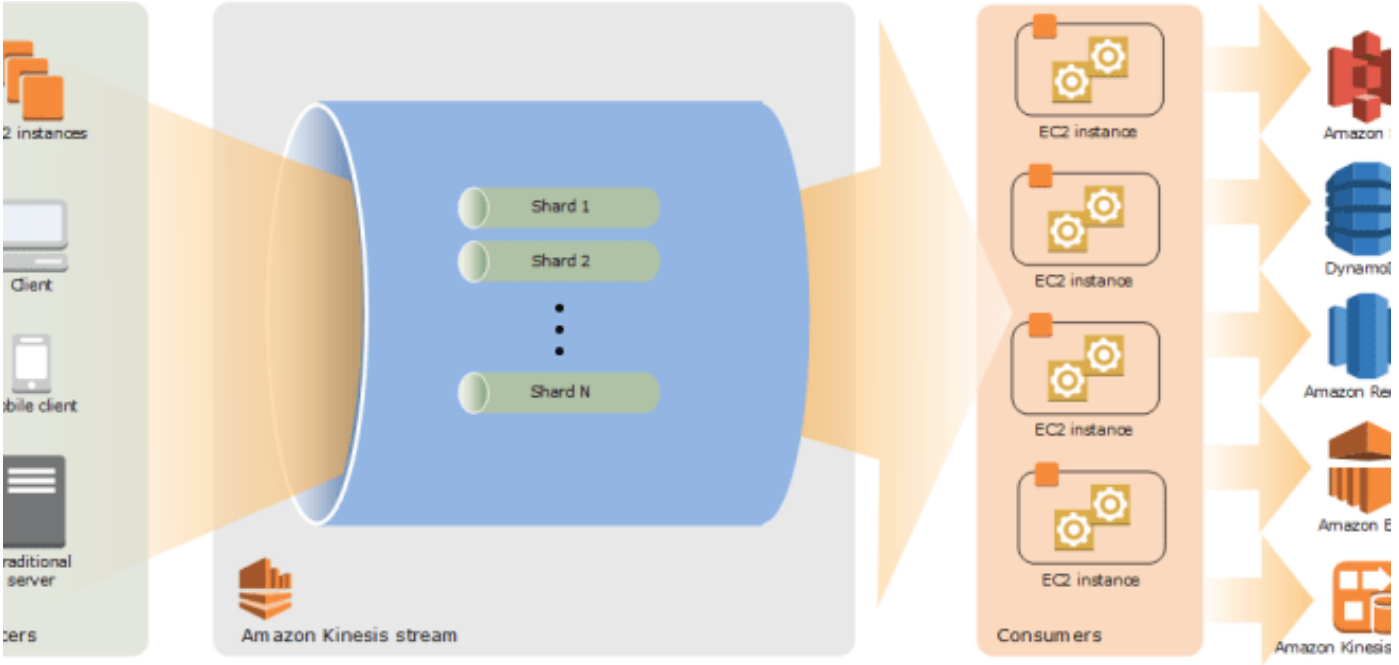
- Kinesis Streams is useful for rapidly moving data off data producers and then continuously processing the data, be it to transform the data before emitting to a data store, run real-time metrics and analytics, or derive more complex data streams for further processing
    - **Accelerated log and data feed intake:** Data producers can push data to Kinesis stream as soon as it is produced, preventing any data loss and making it available for processing within seconds.
    - **Real-time metrics and reporting:** Metrics can be extracted and used to generate reports from data in real-time.
    - **Real-time data analytics**: Run real-time streaming data analytics.
    - **Complex stream processing**: Create Directed Acyclic Graphs (DAGs) of Kinesis Applications and data streams, with Kinesis applications adding to another Amazon Kinesis stream for further processing, enabling successive stages of stream processing.

- **Kinesis limits**
    - stores records of a stream for up to 24 hours, by default, which can be extended to max 7 days
    - maximum size of a data blob (the data payload before Base64-encoding) within one record is 1 megabyte (MB)

- Each shard can support up to 1000 PUT records per second
- Each account can provision 10 shards per region, which can be increased further through request

- **Amazon Kinesis is designed to process streaming big data and the pricing model allows heavy PUTs rate.**

- **Amazon S3 is a cost-effective way to store your data, but not designed to handle a stream of data in real-time**



Kinesis Streams

- Shard
  - **Streams are made of shards and is the base throughput unit of an Kinesis stream.**
  - Each shard provides a capacity of 1MB/sec data input and 2MB/sec data output
  - Each shard can support up to 1000 PUT records per second
  - All data is stored for 24 hours.
  - Replay data inside a 24-hour window
  - Shards define the capacity limits. If the limits are exceeded, either by data throughput or the number of PUT records, the put data call will be rejected with a *ProvisionedThroughputExceeded* exception.
  - This can be handled by
    - Implementing a retry on the data producer side, if this is due to a temporary rise of the stream's input data rate
    - Dynamically scaling the number of shared (resharding) to provide enough capacity for the put data calls to consistently succeed

- Record
  - A record is the unit of data stored in an Amazon Kinesis stream.
  - A record is composed of a sequence number, partition key, and data blob.
  - Data blob is the data of interest your data producer adds to a stream.
  - Maximum size of a data blob (the data payload before Base64-encoding) is 1 MB

- Partition key
  - Partition key is used to segregate and route records to different shards of a stream.
  - A partition key is specified by your data producer while adding data to an Amazon Kinesis stream

- Sequence number
  - A sequence number is a unique identifier for each record.
  - Sequence number is assigned by Amazon Kinesis when a data producer calls PutRecord or PutRecords operation to add data to an Amazon Kinesis stream.
  - Sequence numbers for the same partition key generally increase over time; the longer the time period between PutRecord or PutRecords requests, the larger the sequence numbers become.

Kinesis Streams Components

- Data to an Amazon Kinesis stream can be added via PutRecord and PutRecord operations, , or .
  - Amazon Kinesis Agent

        - is a pre-built Java application that offers an easy way to collect and send data to Amazon Kinesis stream.
        - can be installed on a Linux-based server environments such as web servers, log servers, and database servers
        - can be configured to monitor certain files on the disk and then continuously send new data to the Amazon Kinesis stream

    - Amazon Kinesis Producer Library (KPL)
        - is an easy to use and highly configurable library that helps you put data into an Amazon Kinesis stream.
        - presents a simple, asynchronous, and reliable interface that enables you to quickly achieve high producer throughput with minimal client resources.

- Amazon Kinesis Application is a data consumer that reads and processes data from an Amazon Kinesis stream and can be build using either or

    - Amazon Kinesis Client Library (KCL)

        - is a pre-built library with multiple language support
        - delivers all records for a given partition key to same record processor
        - makes it easier to build multiple applications reading from the same Kinesis stream for e.g. to perform counting, aggregation, and filtering
        - handles complex issues such as adapting to changes in stream volume, load-balancing streaming data, coordinating distributed services, and processing data with fault-tolerance
        - is a pre-built library that helps you easily integrate Amazon Kinesis Streams with other AWS services and third-party tools
        - Kinesis Client Library is required for Kinesis Connector Library
    - is a pre-built library that helps you easily integrate Amazon Kinesis Streams with

Kinesis vs SQS

- **Kinesis Streams enables real-time processing of streaming big data while SQS offers a reliable, highly scalable hosted queue for storing messages and move data between distributed application components**

- Kinesis provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple Amazon Kinesis Applications while SQS does not guarantee data ordering and provides at least once delivery of messages
- Kinesis stores the data up to 24 hours, by default, and can be extended to 7 days while SQS stores the message up to 4 days, by default, and can be configured from 1 minute to 14 days but clears the message once deleted by the consumer

- Kineses and SQS both guarantee at-least once delivery of message

- Kinesis supports multiple consumers while SQS allows the messages to be delivered to only one consumer at a time and requires multiple queues to deliver message to multiple consumers

- **Kinesis use cases requirements**
    - Ordering of records.
    - Ability to consume records in the same order a few hours later
    - Ability for multiple applications to consume the same stream concurrently
    - Routing related records to the same record processor (as in streaming MapReduce)

- **SQS uses cases requirements**
    - Messaging semantics like message-level ack/fail and visibility timeout
    - Leveraging SQS's ability to scale transparently
    - Dynamically increasing concurrency/throughput at read time
    - Individual message delay, which can be delayed

AWS Kinesis - Key notes

- Streaming of data is the data that is generated and sent continuously from a large number of data sources, where data is sent in small sizes

- Kinesis is a platform for streaming data on AWS (used for IoT and Bigdata Analytics:
  - It offers powerful services to make it easy to load an analyze streaming data
  - It facilitates the way to build custom streaming data APPS for specialized needs
  - Kinesis is an AWS managed streaming data service

- Amazon Kinesis enables real-time processing of streaming data at massive scale. It can continuously capture and store Terabytes of data per hour from 100s of thousands of sources

- **Kinesis Use Cases:**
  - IoT sensors data-en
  - Log files from customers of mobile and web apps
  - eCommerce purchases
  - In-game player activities
  - Social media networks
  - Financial trading floors and stock markets
  - Telemetry

- Synchronously replicates data across three facilities in an AWS Region, providing high availability and data durability:

  - Accelerated log and data feed intake:
  - Real-time metrics and reporting:
  - Real-time data analytics:
  - Complex stream processing:

- **Amazon Kinesis is designed to process streaming big data and the pricing model allows heavy PUTs rate.**

- **Amazon S3 is a cost-effective way to store your data, but not designed to handle a stream of data in real-time**

- **Redshift is a columnar analytical data warehouse (analytical databae), It is not designed to be continuously loaded rather a small number of concurrent queries performing large scans of data**

- You can analyze Streaming Data from Amazon Kinesis with Amazon Elastic MapReduce (EMR). AWS has an Amazon EMR connector to Amazon Kinesis which facilitates enabling batch processing of data in Kinesis streams with familiar Hadoop tools and analyzing data in Kinesis streams without

having to write, deploy and maintain any independent stream processing applications.

- **It offers three managed services:**
    - **Kinesis Streams(server side encryption and at rest):**
    -
        - if it is only about **receiving large stream of data to collect(store) and process it in real time (Store** it between 1 (by default)/7days (which is configurable) and availing that to custom applications (Kinesis APPS) on EC2 fleet that will process.

        - Amazon Kinesis Streams (custom data-processing applications) applications read data from an Kinesis Stream as data records
            - These applications use Kinesis Client Library and run on Amazon EC2 instances
            - The processed records can be:
                - Sent to dashboards, Generate alerts, Used to Dynamically change pricing and advertising strategies, Save to a variety of other AWS Services (EMR, DynamoDB or Redshift)

        - Kinesis Streams manages the infrastructure, storage, networking and configuration needed to stream your data at the lever of your data throughput

        - Kinesis Streams synchronously replicate data across three availability zones, providing High Availability anda data durability

        - Kinesis can make huge streams of data available for processing by AWS services or customer applications in less than a second

        - Kinesis streams are formed by:
            - **Producers** (put records in)
            - **Consumers** (get records to process)
            - **Kinesis Streams Application** (consumer that runs on a fleet of EC2 instances)
            - **Shard** (uniquely identified group) and **Record** (data unit stored in)

        - Retention Period (default 24H)/can be extended to 7 days for additional charges
        - **Encryption:**
            - Kinesis Streams can automatically encrypt sensitive data as a producer enters it into a stream

            - Use KMS master keys for encryption

            - To read or write to an encrypted stream, producer and consumer applications must have permission to access the master key
            - **<u>Note</u>: Use server-side encryption incurs KMS costs**

- **Kinesis Firehose (deliver real-time streaming to S3 or Redshift for ex.):**

- If it is only about **capturing/delivering the data** to the AWS services

    - **Kinesis Firehose (encryption using Kinesis streams as source):**
        - if it is only about capturing, transforming the format/data and saving it data to AWS services (Redshift, S3)

- Use Kinesis Firehose to deliver real-time streaming data to destinations such as Amazon S3 and Amazon Redshift. It is the easiest way to load streaming data into AWS

    - Kinesis streams can be used as the source to Kinesis Firehose
    - You can configure Kinesis Firehose to transform your data before delivering it

- Fully managed service for automatically capturing real-time data stream from producers (sources) and delivering them to destinations such as:

    - AWS S3
    - AWS Redshift
    - AWS ElasticSearch Service (ES)
    - and Splunk

- It can batch, compress and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security

- Kinesis Data firehose manages the infrastructure, storage, networking. It also scales elastically without requiring any intervention

- Kinesis Data Firehose synchronously replicate data across three availability zones in an AWS Region, providing High Availability and data durability for the data as it is transported to the destinations

- Store data records for up to 24 hours in case the delivery destination is unavailable

- Kinesis Firehose can invoke AWS Lambda function to transform incoming data before delivering it to destinations

- For **AWS S3 destinations**, streaming data is delivered to an S3 bucket.s3
    - If data transformation is enabled, you can optionally back up data to another Amazon S3 bucket

        - **ElasticSearch case is similar**

    - For **AWS Redshift destinations**, streaming data is delivered to your S3 bucket first and then to is loaded to the Redshift cluster with the Redshift **COPY** command
        - If data transformation is enabled, you can optionally back up data to another Amazon S3 bucket

    - **Encryption:**
        - If you have sensitive data, you can enable server-side data encryption when you use Amazon Kinesis Firehose **but this is only possible if you use a Kinesis Stream as your data source.**

- **Kinesis Analytics (SQL code based):**

    - Use Amazon Kinesis Analytics to process and analyze streaming data with standard SQL
    - Quickly author SQL code that continuosly read, processes and stores data in near real time
    - Supports ingesting data from Kinesis Streams and Kinesis Firehose streaming sources
    - You can also configure destinations where you want to persist the result (S3, Redshift)
    - Supports Kinesis Firehose (S3, Redshift and Elastic Service) or Kinesis streams and Kinesis Streams as destinations
    - Needs permissions to read records from a streaming source and write application output to the external destinations.
        - **Use IAM roles to grant these permissions.**

AWS Simple Email Service – SES

- **SES is a managed service and ideal for sending bulk emails at scale**
- **To send file change notifications via email**
- **SES acts as an outbound email server and eliminates the need to support own software or applications to do the heavy lifting of email transport**
- **Send an email using DKIM with SES allows senders to sign their email messages and ISPs**
- Existing email server can also be configure to send outgoing emails through SES with no change in any settings in the email clients
- Maximum message size including attachments is 10 MB per message

SES Characteristics

- **Compatible with SMTP**
- **Applications can send email using a single API call in many supported languages Java, .Net, PHP, Perl, Ruby, HTTPs etc**
- **Optimized for highest levels of uptime, availability and scales as per the demand**
- **Provides sandbox environment for testing**

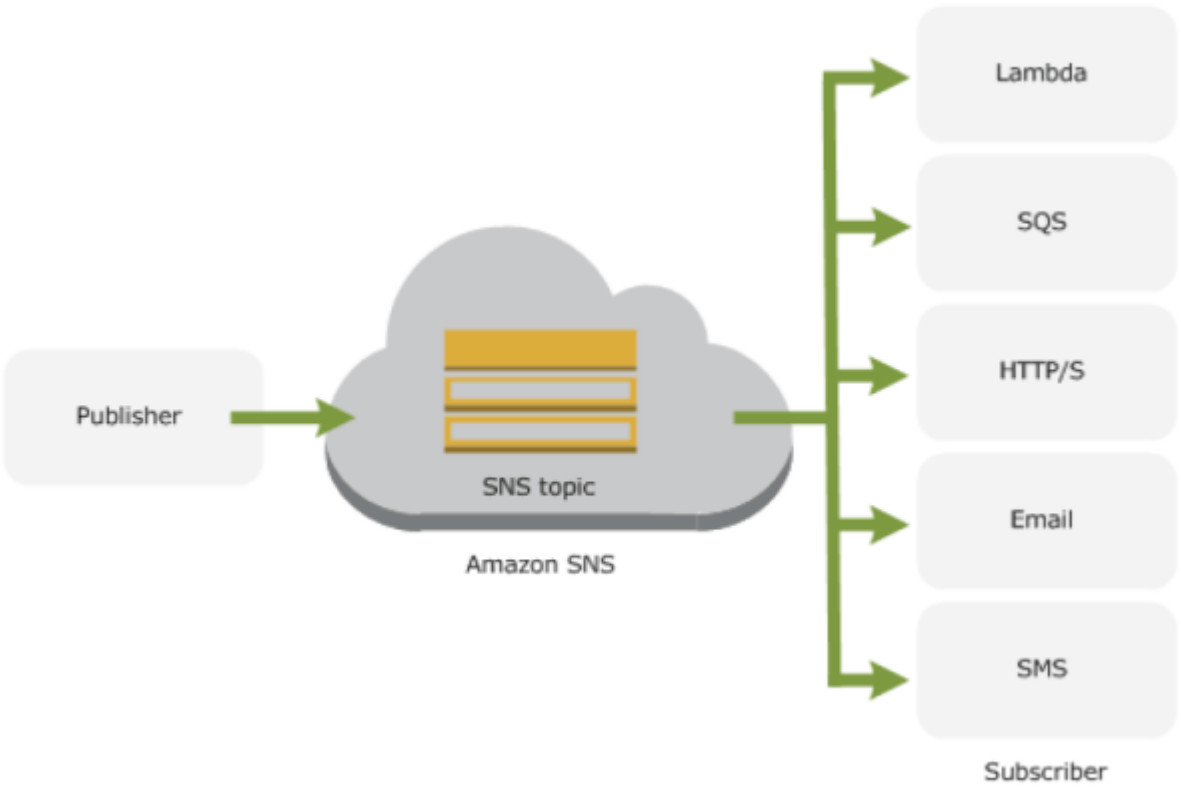
Sending Limits

- Production SES has a set of sending limits which include
    - **Sending Quota** – max number of emails in 24-hour period
    - **Maximum Send Rate** – max number of emails per second
- SES automatically adjusts the limits upward as long as emails are of high quality and they are sent in a controlled manner, as any spike in the email sent might be considered to be spams.
- Limits can also be raised by submitting a Quota increase request

SES Best Practices

- Send high-quality and real production content that your recipients want
- Only send to those who have signed up for the mail
- Unsubscribe recipients who have not interacted with the business recently
- Have low bounce and compliant rates and remove bounced or complained addresses, using SNS to monitor bounces and complaints, treating them as opt-out
- Monitor the sending activity

Simple Notification Service – SNS



- **Simple Notification Service – SNS is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients**
- **SNS provides the ability to create Topic which is a logical access point and communication channel.**
- **Each topic has a unique name that identifies the SNS endpoint for publishers to post messages and subscribers to register for notifications.**
- Producers and Consumers communicate asynchronously with subscribers by producing and sending a message to a topic
- **Producers push messages to the topic, they created or have access to, and SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers**
- Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

- Subscribers (i.e., web servers, email addresses, SQS queues, AWS Lambda functions) consume or receive the message or notification over one of the supported protocols (i.e., SQS, HTTP/S, email, SMS, Lambda) when they are subscribed to the topic.

Accessing Amazon SNS

- **Amazon Management console**
  - Amazon Management console is the web-based user interface which can be used to manage SNS
- **AWS Command line Interface (CLI)**
  - Provides commands for a broad set of AWS products, and is supported on Windows, Mac, and Linux.
- **AWS Tools for Windows Powershell**
  - Provides commands for a broad set of AWS products for those who script in the PowerShell environment
- **AWS SNS Query API (HTTP or HTTPS / GET or POST)**
  - Query API allows for requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named Action
- **AWS SDK libraries**
  - AWS provide libraries in various languages which provide basic functions that automate tasks such as cryptographically signing your requests, retrying requests, and handling error responses

SNS Supported Transport Protocols

- **HTTP, HTTPS** – Subscribers specify a URL as part of the subscription registration; notifications will be delivered through an HTTP POST to the specified URL.

- **Email, Email-JSON** – Messages are sent to registered addresses as email. Email-JSON sends notifications as a JSON object, while Email sends text-based email.

- **SQS** – Users can specify an SQS queue as the endpoint; SNS will enqueue a notification message to the specified queue (which subscribers can then process using SQS APIs such as ReceiveMessage, DeleteMessage, etc.)
  - **Fanout:**
    - Is when an SNS message is sent to a topic and then replicated to multiple SQS queues

- **SMS** – Messages are sent to registered phone numbers as SMS text messages

SNS Supported Endpoints

- **Email Notifications**
  - SNS provides the ability to send Email notifications

- **Mobile Push Notifications**
  - SNS provides an ability to send push notification messages directly to apps on mobile devices. Push notification messages sent to a mobile endpoint can appear in the mobile app as message alerts, badge updates, or even sound alerts
  - Supported push notification services
    - Amazon Device Messaging (ADM)
    - Apple Push Notification Service (APNS)
    - Google Cloud Messaging (GCM)
    - Windows Push Notification Service (WNS) for Windows 8+ and Windows Phone 8.1+
    - Microsoft Push Notification Service (MPNS) for Windows Phone 7+
    - Baidu Cloud Push for Android devices in China

- **SQS Queues**

  - SNS with SQS provides the ability for messages to be delivered to applications that require immediate notification of an event, and also persist in an SQS queue for other applications to process at a later time

  - SNS allows applications to send time-critical messages to multiple subscribers through a “push” mechanism, eliminating the need to periodically check or “poll” for updates.

  - SQS can be used by distributed applications to exchange messages through a polling model, and can be used to decouple sending and receiving components, without requiring each component to be concurrently available.

- **SMS Notifications**

  - SNS provides the ability to send and receive (SMS) notifications to SMS-enabled mobile phones and smart phones

- **HTTP/HTTPS Endpoints**
  - SNS provides the ability to send notification messages to one or more HTTP or HTTPS endpoints. When you subscribe an endpoint to a topic, you can publish a notification to the topic and Amazon SNS sends an HTTP POST request delivering the contents of the notification to the subscribed endpoint

- **Lambda**
  - SNS and Lambda are integrated so Lambda functions can be invoked with SNS notifications.
  - When a message is published to an SNS topic that has a Lambda function subscribed to it, the Lambda function is invoked with the payload of the published message

SNS - Key points to remember

- **Is a push messaging service**
- Instead of polling for new messages, SNS pushes the messages to the subscribers.

- To use SNS, you would need to create a **topic**.

- It consists of:
  - **Publisher (producers)** – which sends a message to a **topic**.
  - **Subscriber (consumer)** – receives the message.

- The messages can be pushed to:
  1. SQS Queues
  2. Lambda functions.
  3. HTTP/HTTPS endpoints.
  4. Mobile Push Notifications (Android & IOS)
  5. SMS
  6. Email notifications

Differences between SQS and SNS

- **SNS** – let's you send messages to subscribers (AWS handles it)
- **SQS** – let's you send messages to a queue, which you would need to poll the queue to process the messages

Amazon Simple Queue Service (SQS)

- **Amazon SQS is a highly available distributed queue system**

- A queue is a temporary repository for messages awaiting for processing and acts as a buffer between the component producer and the consumer

- Amazon SQS
  - offers a reliable, highly-scalable, hosted queue for storing messages in transit between computers
  - provides fault tolerant, loosely coupled, flexibility of distributed components of applications to send & receive without requiring each component to be concurrently available
  - helps build distributed application with decoupled components
  - requires no administrative overhead and little configuration
  - **supports the HTTP over SSL (HTTPS) and Transport Layer Security (TLS) protocols for security**

- **SQS provides two types of Queues – Standard & FIFO**

SQS Standard Queue Features & Key Points

- Redundant infrastructure
  - offers reliable and scalable hosted queues for storing messages
  - is engineered to always be available and deliver messages
  - provides ability to store messages in a fail safe queue
  - highly concurrent access to messages

- At-Least-Once delivery
  - ensures delivery of each message at least once
  - stores copies of the messages on multiple servers for redundancy and high availability
  - might deliver duplicate copy of messages, if the servers storing a copy of a message is unavailable when you receive or delete the message and the copy of the message is not deleted on that unavailable server
  - Applications should be designed to be idempotent with the ability to handle duplicate messages and not be adversely affected if it processes the same message more than once

- Message Attributes
  - SQS message can contain up to 10 metadata attributes.
  - take the form of name-type-value triples
  - can be used to separate the body of a message from the metadata that describes it.
  - helps process and store information with greater speed and efficiency because the applications don’t have to inspect an entire message before understanding how to process it

- Message Sample
    - behaviour of retrieving messages from the queue depends on whether short (standard) polling, the default behaviour, or long polling is used
    - **With short polling**,
        - SQS samples only a subset of the servers (based on a weighted random distribution) and returns messages from just those servers.
        - A receive request might not return all the messages in the queue. But a subsequent receive request would return the message
    - **With Long polling,**
        - request persists for the time specified and returns as soon as the message is available thereby reducing costs and time the message has to dwell in the queue

- Batching
    - SQS allows send, receive and delete batching which helps club up to 10 messages in a single batch while charging price for a single message
    - helps lower cost and also increases the throughput

- Configurable settings per queue
    - All queues don’t have to be alike

- Order
    - makes a best effort to preserve order in messages does not guarantee first in, first out delivery of messages
    - can be handled by placing sequencing information within the message and performing the ordering on the client side

- Loose coupling
    - removes tight coupling between components
    - provides the ability to move data between distributed components of the applications that perform different tasks without losing messages or requiring each component to be always available

- Multiple writers and readers
    - supports multiple readers and writers interacting with the same queue as the same time
    - locks the message during processing, using Visibility Timeout, preventing it to be processed by any other consumer

- Variable message size
    - supports message in any format up to 256KB of text.
    - messages larger than 256 KB can be managed using the S3 or DynamoDB, with SQS holding a pointer to the S3 object

- Access Control
    - Access can be controlled for who can produce and consume messages to each queue

- Delay Queues
    - delay queue allows the user to set a default delay on a queue such that delivery of all messages enqueued is postponed for that time duration

- Dead Letter Queues
    - **Dead letter queue is a queue for messages that were not able to be processed after a maximum number of attempts**

- PCI Compliance
    - **supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being PCI-DSS (Payment Card Industry – Data Security Standard) compliant**

SQS Use Cases

- Work Queues
    - Decouple components of a distributed application that may not all process the same amount of work simultaneously.

- Buffer and Batch Operations
    - Add scalability and reliability to the architecture and smooth out temporary volume spikes without losing messages or increasing latency

- Request Offloading
    - Move slow operations off of interactive request paths by en-queueing the request.

- Fan-out
    - Combine SQS with SNS to send identical copies of a message to multiple queues in parallel for simultaneous processing.

- Auto Scaling
    - SQS queues can be used to determine the load on an application, and combined with Auto Scaling, the EC2 instances can be scaled in or out, depending on the volume of traffic

• How SQS Queues Works

- SQS allows queues to be created, deleted and messages can be sent and received from it
- SQS queue retains messages for four days, by default.
- Queues can configured to retain messages for 1 minute to 14 days after the message has been sent.
- SQS can delete a queue without notification if one of the following actions hasn't been performed on it for 30 consecutive days.
- SQS allows the deletion of the queue with messages in it

Queue and Message Identifiers

Queue URLs

- **Queue is identified by a unique queue name within the same AWS account**
- SQS assigns each queue with a Queue URL identifier for e.g. http://sqs.us-east-1.amazonaws.com/123456789012/queue2
- **Queue URL is needed to perform any operation on the Queue**

Message ID

- Message IDs are useful for identifying messages,
- Each message receives a system-assigned message ID that SQS returns to with the SendMessage response.
- **To delete a message, the message's receipt handle instead of the message ID is needed**
- Message ID can be of is 100 characters max

Receipt Handle

- **When a message is received from a queue, a receipt handle is returned** with the message which is associated with the act of receiving the message rather then the message itself
- Receipt handle is required, not the message id, to delete a message or to change the message visibility
- If a message is received more than once, each time its received, a different receipt handle is assigned and the latest should be used always

Visibility time-out

- Behaviour
    - SQS does not delete the message once it is received by a consumer, because the system is distributed, there's no guarantee that the consumer will actually receive the message (it's possible the connection could break or the component could fail before receiving the message)

    - Consumer should explicitly delete the message from the Queue once it is received and successfully processed

        - As the message is still available on the Queue, other consumers would be able to receive and process and this needs to be prevented

- SQS handles the above behaviour using Visibility timeout.

- SQS blocks the visibility of the message for the Visibility time-out period, which is the time during which SQS prevents other consuming components from receiving and processing that message

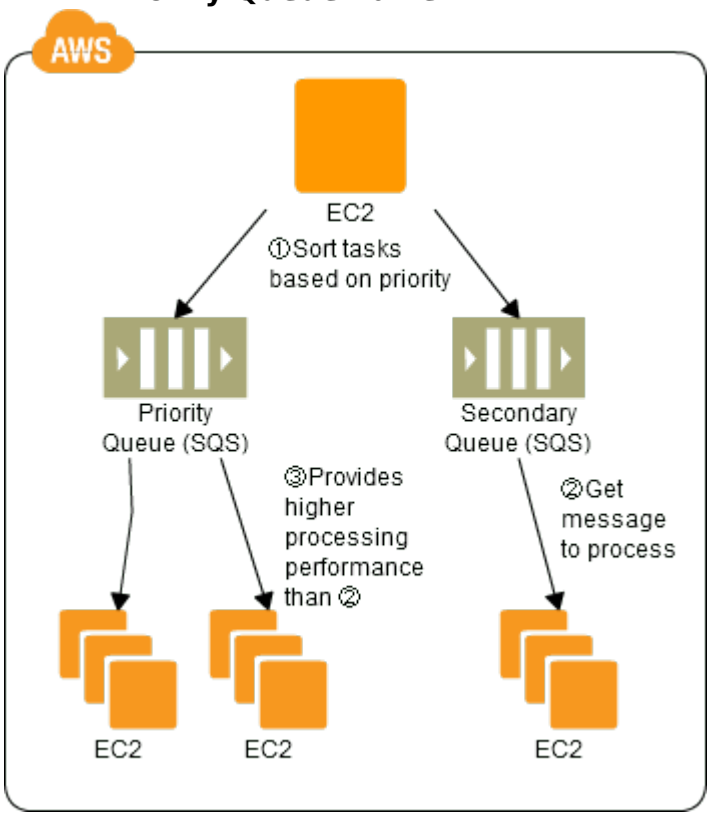
- Consumer should delete the message within the Visibility time-out. If the consumer fails to delete the message before the visibility time-out expires, the message is visible again for other consumers.

- **Visibility time-out considerations**
    - clock starts ticking once SQS returns the message
    - should be large enough to take into account the processing time for each of the message
    - default Visibility time-out for each Queue is 30 seconds and can be changed at the Queue level
    - when receiving messages, a special visibility timeout for the returned messages can be set without changing the overall queue time-out using the receipt handle
    - can be extended by the consumer, if the consumer thinks it won't be able to process the message within the current visibility time-out period. SQS restarts the time-out period using the new value
    - a message's Visibility time-out extension applies only to that particular receipt of the message and does not affect the time-out for the queue or later receipts of the message

- SQS has an 120,000 limit for the number of in-flight messages per queue i.e. message received but not yet deleted and any further messages would receive an error after reaching the limit

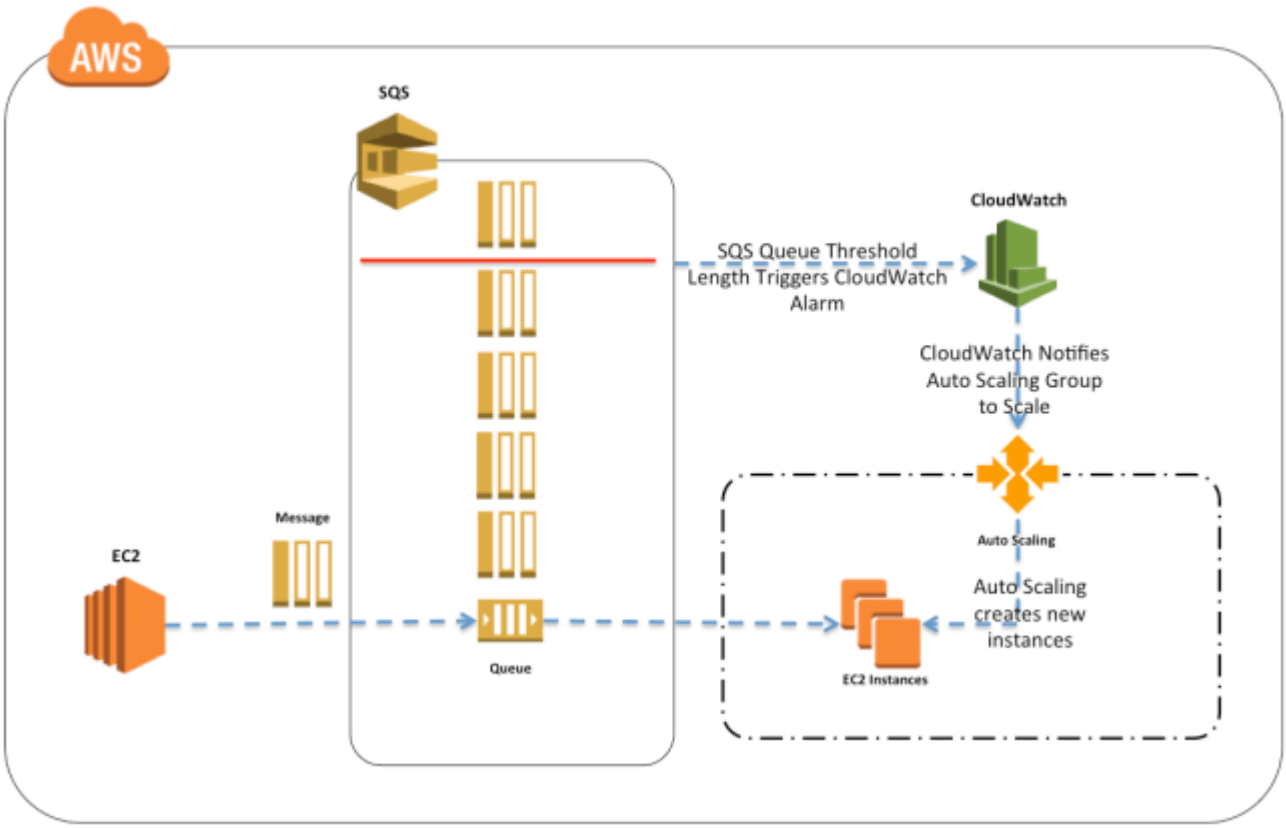
SQS Design Patterns

- **Priority Queue Pattern**



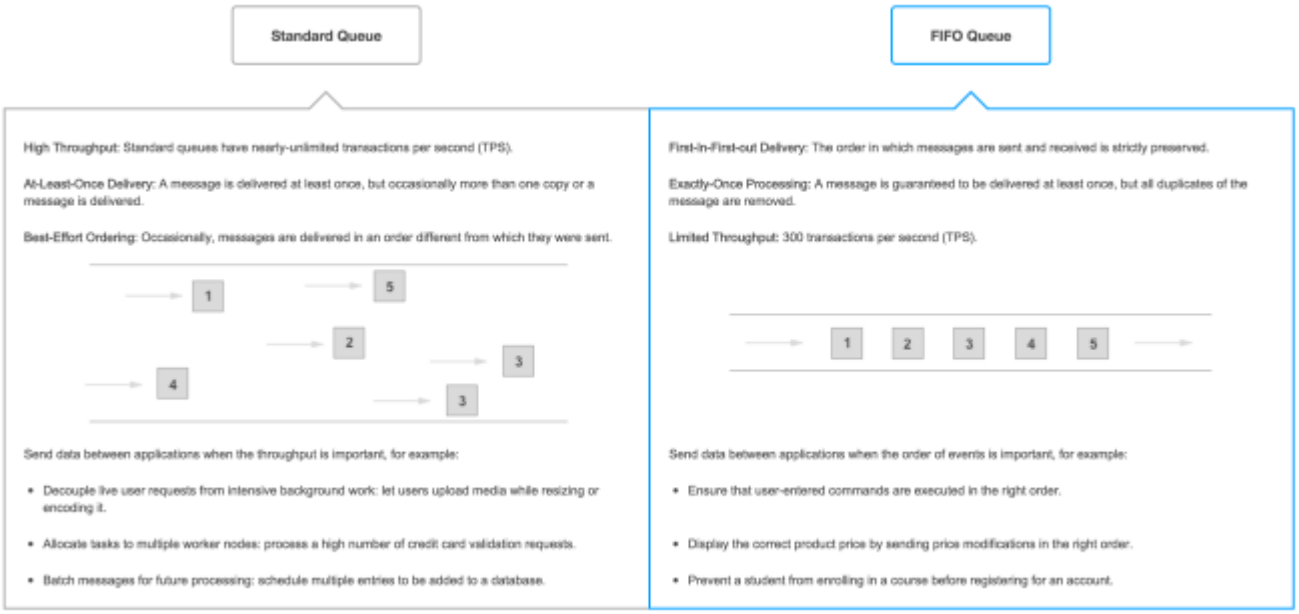
1. Use SQS to prepare multiple queues for the individual priority levels.
2. Place those processes to be executed immediately (job requests) in the high priority queue.
3. Prepare numbers of batch servers, for processing the job requests of the queues, depending on the priority levels.
4. Queues have a message “Delayed Send” function, which can be used to delay the time for starting a process.

SQS Job Observer Pattern



1. Enqueue job requests as SQS messages.
2. Have the batch server dequeue and process messages from SQS.
3. Set up Auto Scaling to automatically increase or decrease the number of batch servers, using the number of SQS messages, with CloudWatch, as the trigger to do so.

AWS SQS – Standard vs FIFO Queue



Message Order

- Standard queues provide best-effort ordering which ensures that messages are generally delivered in the same order as they are sent. Occasionally (because of the highly-distributed architecture that allows high throughput), more than one copy of a message might be delivered out of order
- FIFO queues offer first-in-first-out delivery and exactly-once processing: the order in which messages are sent and received is strictly preserved

Delivery

- Standard queues guarantee that a message is delivered at least once and duplicates can be introduced into the queue
- FIFO queues ensure a message is delivered exactly once and remains available until a consumer processes and deletes it; duplicates are not introduced into the queue

Transactions Per Second (TPS)

- Standard queues allow nearly-unlimited number of transactions per second
- FIFO queues are limited to 300 transactions per second per API action

Regions

- Standard queues are available in all the regions
- FIFO queues are currently available in limited regions including the US West (Oregon), US East (Ohio), US East (N. Virginia), and EU (Ireland)

SQS Buffered Asynchronous Client

- FIFO queues aren’t currently compatible with the SQS Buffered Asynchronous Client, where messages are buffered at client side and send as a single request to the SQS queue to reduce cost.

AWS Services Supported

- **Standard Queues are supported by all AWS services**

- FIFO Queues are currently not supported by all AWS services like
    - CloudWatch Events
    - S3 Event Notifications
    - SNS Topic Subscriptions
    - Auto Scaling Lifecycle Hooks
    - AWS IoT Rule Actions
    - AWS Lambda Dead Letter Queues

Use Cases (Standard VS FIFO queues)

- **Standard queues can be used in any scenarios, as long as the application can process messages that arrive more than once and out of order**
    - Decouple live user requests from intensive background work: Let users upload media while resizing or encoding it.
    - Allocate tasks to multiple worker nodes: Process a high number of credit card validation requests.
    - Batch messages for future processing: Schedule multiple entries to be added to a database.

- **FIFO queues are designed to enhance messaging between applications when the order of operations and events is critical**, **or where duplicates can’t be tolerated**
    - Ensure that user-entered commands are executed in the right order.
    - Display the correct product price by sending price modifications in the right order.
    - Prevent a student from enrolling in a course before registering for an account.

SQS - Key points to remember

What is SQS?
A best practice when building scalable and highly available systems on AWS is to decouple your micro-service by using one of the following options:

- **ELB (Load Balancer) for synchronous decoupling:** web application answering HTTPS requests from browsers

- **SQS queue for asynchronous decoupling: sending out massive amounts of emails, transcoding video files after upload, or analyzing user behaviour.**

    - Amazon Simple Queue Service (SQS) is a managed message queuing service. SQS offers fault tolerant and scalable distributed message queues: simple to use but very powerful.

    - The number of consumers can be scaled based on the number of tasks in the queue. As the tasks are stored durable and are managed within SQS, it is also very easy to build a fault tolerant system recovering from failed consumers without losing any tasks automatically.

    - **SQS offers a REST API**, so there are various options to integrate into producers and consumers: mobile or client-side web applications, applications running on EC2 instances, and AWS Lambda.

    - **supports the HTTP over SSL** (HTTPS) and Transport Layer Security (TLS) protocols for security
      PCI Compliance: supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being PCI-DSS (Payment Card Industry – Data Security Standard) compliant

    - **Fan-out:** Combine SQS with SNS to send identical copies of a message to multiple queues in parallel for simultaneous processing.

    - When a message is received from a queue, a receipt handle is returned

SQS: Simple Queue Service

- It is used to decouple components **(note: in the exam whenever you are asked how would you decouple an architecture it is always SQS)**
- **Example use case:**
    - **To reduce the insert rate into the database**. Data can be sent to the Queue and a process could insert the data from the queue into the database.

    - **To reduce the delay in creating SMTP connectio**n, data can be sent to the Queue.
- Minimum size of message can be 1 byte (1 character). Maximum size of a message can be 256 KB.
- Queue name can be up to 80 characters.
- Messages can contain up to 10 meta-data attributes.
- **Dead letter queue is a queue for messages that were not able to be processed after a maximum number of attempts .** It is important to keep the Dead Letter Queue empty as much as possible!

SQS Types

There are two types of SQS:

1. **Standard queues:**
    - **Standard queues can be used in any scenarios, as long as the application can process messages that arrive more than once and out of order**
    - At-least one delivery
    - Unlimited number of transaction.
    - Not in any order.

2. **FIFO queues (First in First Out):**
    - **FIFO queues are designed to enhance messaging between applications when the order of operations and events is critical, or where duplicates can't be tolerated**
    - No duplicates.
    - Messages are in order compared to standard queues.
    - First queue support up to 300 messages per second.

Delay Queues

- The delivery of new messages in a queue.
- Messages delivered in the queue are not accessible.
- The delay can be set when a queue is created using the SetQueueAttributes.

Queue Identifiers

SQS has three identifiers:

1. **URL**: AWS assigns each queue a unique URL.
2. **Unique ID**: AWS assigns each message a unique ID used to identify the message.
3. **Receipt Handles:** On every message received from a queue, a receipt handle is returned.

    "To delete a message, you would need the message receipt handle not the Message ID"

- **Long Polling**
    - Long polling is a process of checking for messages to process in the queue.

    - **request persists for the time specified and returns as soon as the message is available thereby reducing costs and time the message**
        - Consider Long polling a while loop that checks for messages in the queue. However, a while loop that is continuously calling SQS for messages is inefficient and burns CPU cycle.
        - To save CPU cycle and be efficient, instead of checking the SQS continuously, we could send a WaitTimeSeconds to receive messages up to a (between 0 and 20 seconds)
        - **In the exam, you will be asked how to make long polling efficient, select the option which has WaitTimeSeconds.**
    - Messaging queue that handles messages or work flows between components
    - supports unlimited number of queues and messages but automatically delete messages in the queue for more than 4 days
    - behaviour of retrieving messages from the queue depends on whether short (standard) polling, the default behaviour, or long polling is used

- **Short polling**,
    - SQS samples only a subset of the servers (based on a weighted random distribution) and returns messages from just those servers.
    - A receive request might not return all the messages in the queue. But a subsequent receive request would return the message

AWS SWF – Simple WorkFlow service

- **AWS SWF makes it easy to build applications that coordinate work across distributed components**

- SWF makes it easier to develop asynchronous and distributed applications by providing a programming model and infrastructure for coordinating distributed components, tracking and maintaining their execution state in a reliable way

- **SWF does the following**
  - stores metadata about a workflow and its component parts.
  - stores task for workers and queues them until a Worker needs them.
  - assigns task to workers, which can run either on cloud or on-premises
  - routes information between executions of a workflow and the associated Workers.
  - tracks the progress of workers on Tasks, with configurable timeouts.
  - maintains workflow state in a durable fashion

- SWF helps coordinating tasks across the application which involves managing intertask dependencies, scheduling, and concurrency in accordance with the logical flow of the application.

- SWF gives full control over implementing tasks and coordinating them without worrying about underlying complexities such as tracking their progress and maintaining their state.

- SWF tracks and maintains the workflow state in a durable fashion, so that the application is resilient to failures in individual components, which can be implemented, deployed, scaled, and modified independently

- **SWF offers capabilities to support a variety of application requirements and is suitable for a range of use cases that require coordination of tasks, including media processing, web application back-ends, business process workflows, and analytics pipelines.**

Simple Workflow Concepts

- Workflow
  - Fundamental concept in SWF is the Workflow, which is the automation of a business process
  - A workflow is a set of activities that carry out some objective, together with logic that coordinates the activities.

- Workflow Execution
    - A workflow execution is a running instance of a workflow

- Workflow History
    - SWF maintains the state and progress of each workflow execution in its Workflow History, which saves the application from having to store the state in a durable way.
    - It enables applications to be stateless as all information about a workflow execution is stored in its workflow history.
    - For each workflow execution, the history provides a record of which activities were scheduled, their current status, and their results. The workflow execution uses this information to determine next steps.
    - History provides a detailed audit trail that can be used to monitor running workflow executions and verify completed workflow executions.
    - Operations that do not change the state of the workflow for e.g. polling execution do not typically appear in the workflow history
    - Markers can be used to record information in the workflow history of a workflow execution that is specific to the use case

- Domain
    - Each workflow runs in an AWS resource called a Domain, which controls the workflow's scope
    - An AWS account can have multiple domains, with each containing multiple workflows
    - **Workflows in different domains cannot interact with each other**

- Activities
    - Designing an SWF workflow, Activities need to be precisely defined and then registered with SWF as an activity type with information such as name, version and time-out

- Activity Task & Activity Worker
    - An Activity Worker is a program that receives activity tasks, performs them, and provides results back. An activity worker can be a program or even a person who performs the task using an activity worker software

    - Activity tasks—and the activity workers that perform them can
        - run synchronously or asynchronously, can be distributed across multiple computers, potentially in different geographic regions, or run on the same computer,
        - be written in different programming languages and run on different operating systems
        - be created that are long-running, or that may fail, time out require restarts or that may complete with varying throughput & latency

- Decider
    - A Decider implements a Workflow's coordination logic.

    - **Decider schedules activity tasks, provides input data to the activity workers, processes events that arrive while the workflow is in progress, and ends (or closes) the workflow when the objective has been completed.**
    - **Decider directs the workflow by receiving decision tasks from SWF and responding back to SWF with decisions.** A decision represents an action or set of actions which are the next steps in the workflow which can either be to schedule an activity task, set timers to delay the execution of an activity task, to request cancellation of activity tasks already in progress, and to complete or close the workflow.

- **Workers and Deciders are both stateless**, and can respond to increased traffic by simply adding additional Workers and Deciders as needed

- Role of SWF service is to function as a reliable central hub through which data is exchanged between the decider, the activity workers, and other relevant entities such as the person administering the workflow.

- Mechanism by which both the activity workers and the decider receive their tasks (activity tasks and decision tasks resp.) is by polling the SWF

- SWF allows “long polling”, requests will be held open for up to 60 seconds if necessary, to reduce network traffic and unnecessary processing

- SWF informs the decider of the state of the workflow by including with each decision task, a copy of the current workflow execution history. The workflow execution history is composed of events, where an event represents a significant change in the state of the workflow execution for e.g events would be the completion of a task, notification that a task has timed out, or the expiration of a timer that was set earlier in the workflow execution. The history is a complete, consistent, and authoritative record of the workflow’s progress

Workflow Implementation & Execution

1. Implement Activity workers with the processing steps in the Workflow.
2. Implement Decider with the coordination logic of the Workflow.
3. Register the Activities and workflow with SWF.
4. Start the Activity workers and Decider. Once started, the decider and activity workers should start polling Amazon SWF for tasks.
5. Start one or more executions of the Workflow. Each execution runs independently and can be provided with its own set of input data.
6. When an execution is started, SWF schedules the initial decision task. In response, the decider begins generating decisions which initiate activity tasks. Execution continues until your decider makes a decision to close the execution.
7. View and Track workflow executions

SWF – Key notes

- **AWS SWF makes it easy to build applications that coordinate work across distributed components**

- **SWF is used to coordinate tasks across distributed components.**

- SWF is in Video Encoding, Data Centre Migration, e-commerce websites (Amazon).

Workflows

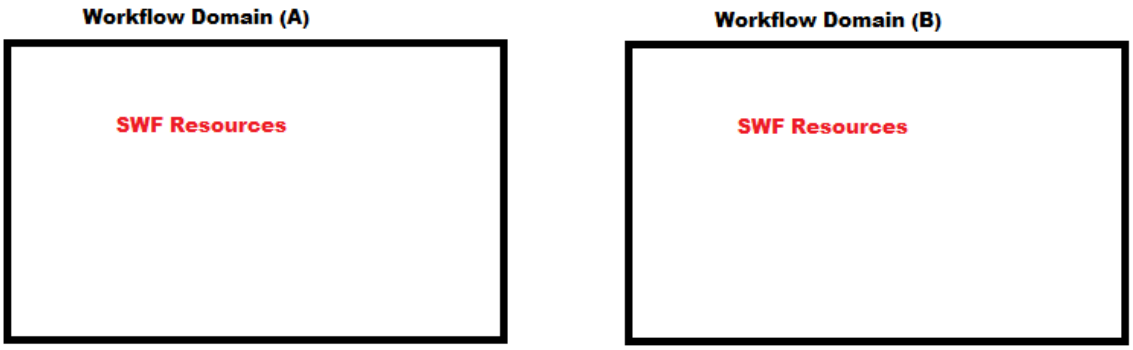
- **It is the automation of a business process**
- Collection of activities.
- **Manages the order in which activities are executed**

Example: Let's say on amazon.co.uk , it is the worker that receives the order of a customer.

Workflow Domain:

- A domain must be provided for all the components of a workflow.
- Can have more than one workflow in a domain.
- **Workflows in different domains cannot interact with each other.**

Consider workflow domain as a folder in which you place all your SWF resources. But Workflow in the folder cannot interact with another folders.



Actors

Actors can be:

- Starters.
- Deciders.
- Activity Workers.

Starter
Initiates the workflow executions.
(you might receive a question; which actor is responsible for initiating workflow? Answer is **Starter**)

Deciders
Is responsible for making decision on workflow execution.
Providers the input data to the activity workers.

Activity Workers
Performs the activity tasks in the workflow.
Polls for new tasks to be work on.

Amazon Elastic MapReduce (EMR)

- **Amazon EMR is a web service that utilizes a hosted Hadoop framework running on the web-scale infrastructure of EC2 and S3**

- EMR enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data

- **EMR**
  - uses Apache Hadoop as its distributed data processing engine, which is an open source, Java software that supports data-intensive distributed applications running on large clusters of commodity hardware
  - is ideal for problems that necessitate fast and efficient processing of large amounts of data
  - lets the focus be on crunching or analyzing big data without having to worry about time-consuming set-up, management or tuning of Hadoop clusters or the compute capacity
  - can help perform data-intensive tasks for applications such as web indexing, data mining, log file analysis, machine learning, financial analysis, scientific simulation, and bioinformatics research etc.
  - provides web service interface to launch the clusters and monitor processing-intensive computation on clusters
  - is a batch-processing framework that measures the common processing time duration in hours to days, **if the use case is to have processing at real time or within minutes Apache Spark or Storm would be a better option**

- **EMR seamlessly supports On-Demand, Spot, and Reserved Instances**

- **EMR launches all nodes for a given cluster in the same EC2 Availability Zone, which improves performance as it provides higher data access rate**

- **EMR supports different EC2 instance types including Standard, High CPU, High Memory, Cluster Compute, High I/O, and High Storage**
    - Standard Instances have memory to CPU ratios suitable for most general-purpose applications.
    - High CPU instances have proportionally more CPU resources than memory (RAM) and are well suited for compute-intensive applications
    - High Memory instances offer large memory sizes for high throughput applications
    - Cluster Compute instances have proportionally high CPU with increased network performance and are well suited for High Performance Compute (HPC) applications and other demanding network-bound applications
    - High Storage instances offer 48 TB of storage across 24 disks and are ideal for applications that require sequential access to very large data sets such as data warehousing and log processing

- **EMR charges on hourly increments i.e. once the cluster is running, charges apply entire hour**

- **EMR integrates with CloudTrail to record AWS API calls**
  NOTE: Topic mainly for Solution Architect Professional Exam Only

EMR Architecture

- Amazon EMR uses industry proven, fault-tolerant Hadoop software as its data processing engine

- Hadoop is an open source, Java software that supports data-intensive distributed applications running on large clusters of commodity hardware

- Hadoop splits the data into multiple subsets and assigns each subset to more than one EC2 instance. So, if an EC2 instance fails to process one subset of data, the results of another Amazon EC2 instance can be used

- **EMR consists of Master node, one or more Slave nodes**
    - **Master Node**
        - EMR currently does not support automatic fail-over of the master nodes or master node state recovery
        - If master node goes down, the EMR cluster will be terminated and the job needs to be re-executed
    - **Slave Nodes – Core nodes and Task nodes**
        - Core nodes
            - host persistent data using Hadoop Distributed File System (HDFS) and run Hadoop tasks
            - can be increased in an existing cluster
        - Task nodes
            - only run Hadoop tasks
            - can be increased or decreased in an existing cluster

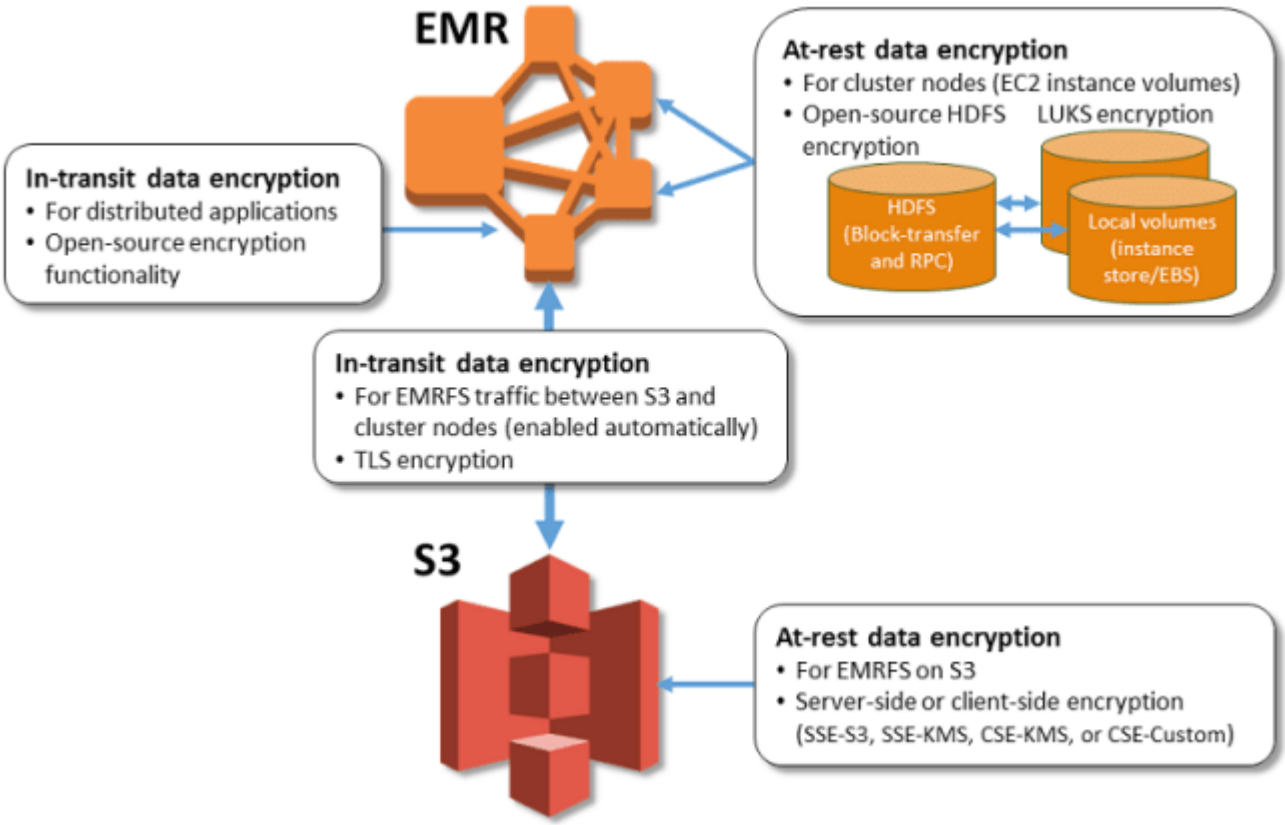
- EMR is fault tolerant for slave failures and continues job execution if a slave node goes down.
- **Currently, EMR does not automatically provision another node to take over failed slaves**

- **EMR supports Bootstrap actions which allow**
  - users a way to run custom set-up prior to the execution of the cluster.
  - can be used to install software or configure instances before running the cluster

EMR Security

- EMR cluster starts with different security groups for Master and Slaves
  - Master security group
    - has a port open for communication with the service.
    - has a SSH port open to allow direct SSH into the instances, using the key specified at start-up

  - Slave security group
    - only allows interaction with the master instance
    - SSH to the slave nodes can be done by doing SSH to the master node and then to the slave node
  - Security groups can be configured with different access rules



- **EMR enables use of security configuration**
  - **which helps to encrypt data at-rest, data in-transit, or both**

    - can be used to specify settings for S3 encryption with EMR file system (EMRFS), local disk encryption, and in-transit encryption
    - is stored in EMR rather than the cluster configuration making it reusable
    - gives flexibility to choose from several options, including keys managed by AWS KMS, keys managed by S3, and keys and certificates from custom providers that you supply

- **At-rest Encryption for S3 with EMRFS**
    - EMRFS supports Server-side (SSE-S3, SSE-KMS) and Client-side encryption (CSE-KMS or CSE-Custom)
    - S3 SSE and CSE encryption with EMRFS are mutually exclusive; either one can be selected but not both
    - Transport layer security (TLS) encrypts EMRFS objects in-transit between EMR cluster nodes & S3

- **At-rest Encryption for Local Disks**
    - Open-source HDFS Encryption
        - HDFS exchanges data between cluster instances during distributed processing, and also reads from and writes data to instance store volumes and the EBS volumes attached to instances
        - **Open-source Hadoop encryption options are activated**
            - **Secure Hadoop RPC is set to “Privacy”**, which uses Simple Authentication Security Layer (SASL).
            - Data encryption on HDFS block data transfer is set to true and is configured to use AES 256 encryption.
    - LUKS. In addition to HDFS encryption, the Amazon EC2 instance store volumes (except boot volumes) and the attached Amazon EBS volumes of cluster instances are encrypted using LUKS

- **In-Transit Data Encryption**
    - Encryption artefacts used for in-transit encryption in one of two ways:
        - either by providing a zipped file of certificates that you upload to S3,
        - or by referencing a custom Java class that provides encryption artefacts

• EMR Cluster Types

- **EMR has two cluster types, transient and persistent**

- **Transient EMR Clusters (shut down when the job is done)**
  - Transient EMR clusters are clusters that shut down when the job or the steps (series of jobs) are complete
  - Transient EMT clusters can be used in situations
    - where total number of EMR processing hours per day < 24 hours and its beneficial to shut down the cluster when it's not being used.
    - using HDFS as your primary data storage.
    - job processing is intensive, iterative data processing.

- **Persistent EMR Clusters (continues running after the job is completed)**
  - Persistent EMR clusters continue to run after the data processing job is complete
  - Persistent EMR clusters can be used in situations
    - frequently run processing jobs where it's beneficial to keep the cluster running after the previous job.
    - processing jobs have an input-output dependency on one another.
    - In rare cases when it is more cost effective to store the data on HDFS instead of S3
    -

EMR Best Practices

- **Data Migration**
  - Two tools – S3DistCp and DistCp – can be used to move data stored on the local (data center) HDFS storage to S3, from S3 to HDFS and between S3 and local disk (non HDFS) to S3
  - AWS Import/Export and Direct Connect can also be considered for moving data

- **Data Collection**
  - Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, & moving large amounts of log data
  - Flume agents can be installed on the data sources (web-servers, app servers, etc) and data shipped to the collectors which can then be stored in persistent storage like S3 or HDFS

- **Data Aggregation**
  - Data aggregation refers to techniques for gathering individual data records (for e.g. log records) and combining them into a large bundle of data files i.e. creating a large file from small files
  - Hadoop, on which EMR runs, generally performs better with fewer large files compared to many small files

        - Hadoop splits the file on HDFS on multiple nodes, while for the data in S3 it uses the HTTP Range header query to split the files which helps improve performance by supporting parallelization
        - Log collectors like Flume and Fluentd can be used to aggregate data before copying it to the final destination (S3 or HDFS)
        - Data aggregation has following benefits
            - Improves data ingest scalability by reducing the number of times needed to upload data to AWS
            - Reduces the number of files stored on S3 (or HDFS), which inherently helps provide better performance when processing data
            - Provides a better compression ratio as compressing large, highly compressible files is often more effective than compressing a large number of smaller files.

- **Data compression**
    - Data compression can be used at the input as well as intermediate outputs from the mappers
    - Data compression helps
        - Lower storage costs
        - Lower bandwidth cost for data transfer
        - Better data processing performance by moving less data between data storage location, mappers, and reducers
        - Better data processing performance by compressing the data that EMR writes to disk, i.e. achieving better performance by writing to disk less frequently
    - Data Compression can have an impact on Hadoop data splitting logic as some of the compression techniques like gzip do not support it

- **Data Partitioning**
    - Data partitioning helps in data optimizations and lets you create unique buckets of data and eliminate the need for a data processing job to read the entire data set
    - Data can be partitioned by
        - Data type (time series)
        - Data processing frequency (per hour, per day, etc.)
        - Data access and query pattern (query on time vs. query on geo location)

- **Cost Optimization**
    - AWS offers different pricing models for EC2 instances
        - **On-Demand instances**
            - are a good option if using transient EMR jobs or if the EMR hourly usage is less than 17% of the time
        - **Reserved instances**
            - are a good option for persistent EMR cluster or if the EMR hourly usage is more than 17% of the time as is more cost effective

        - **Spot instances**

        - can be a cost effective mechanism to add compute capacity
        - **can be used where the data is persists on S3**
        - **can be used to add extra task capacity with Task nodes, and**
        - **is not suited for Master node,** as if it is lost the cluster is lost and Core nodes (data nodes) as they host data and if lost needs to be recovered to rebalance the HDFS cluster

    - **Architecture pattern can be used,**
        - Run master node on On-Demand or Reserved Instances (if running persistent EMR clusters).
        - Run a portion of the EMR cluster on core nodes using On-Demand or Reserved Instances and
        - the rest of the cluster on task nodes using Spot Instances.

EMR – S3 vs HDFS

- Storing data on S3 provides several benefits:

    - inherent features, high availability, durability, life-cycle management, data encryption and archival of data to Glacier
    - cost effective as storing data in S3 is cheaper as compared to HDFS with the replication factor
    - ability to use Transient EMR cluster and shutdown the clusters after the job is completed, with data being maintained in S3
    - ability to use Spot instances and not having to worry about losing the spot instances any time
    - provides data durability from any HDFS node failures, where node failures exceed the HDFS replication factor
    - data ingestion with high throughput data stream to S3 is much easier than ingesting to HDFS

EMR Key notes

- **Amazon EMR is a web service that utilizes a hosted Hadoop framework running on the web-scale infrastructure of EC2 and S3**

- It is a batch-processing framework that measures the common processing time duration in hours to days, if the use case is to have processing at real time or within minutes Apache Spark or Storm would be a better option

- **EMR supports:**
  - On-Demand **(transient EMR jobs)**
  - Spot **(cost effective to extra task nodes if data is persistent on S3, not valid for Master node)**
  - Reserved Instances **(good if the usage is mote than 17% time)**

- EMR launches all nodes for a given cluster in the same EC2 Availability Zone, which improves performance as it provides higher data access rate

- EMR supports different EC2 instance types including Standard, High CPU, High Memory, Cluster Compute, High I/O, and High Storage

- EMR charges on hourly increments i.e. once the cluster is running, charges apply entire hour

- EMR integrates with CloudTrail to record AWS API calls

- **EMR Security:**

  - **Master nodes** (has a security group, hadoop port + ssh)
  - **Slave nodes** (security group SSH from master node)
    - **Core nodes** (hot persistent data)
    - **Task nodes** (only run hadoop tasks)

  - EMR enables use of security configuration

    - which helps to encrypt data at-rest, data in-transit, or both
    - can be used to specify settings for S3 encryption with EMR file system (EMRFS), local disk encryption, and in-transit encryption
    - is stored in EMR rather than the cluster configuration making it reusable
    - gives flexibility to choose from several options, including keys managed by AWS KMS, keys managed by S3, and keys and certificates from custom providers that you supply
    - **At-rest Encryption for S3 with EMRFS**
      - EMRFS supports Server-side (SSE-S3, SSE-KMS) and Client-side encryption (CSE-KMS or CSE-Custom)
      - S3 SSE and CSE encryption with EMRFS are mutually exclusive; either one can be selected but not both
    - **In-Transit encryption:**

        - Transport layer security (TLS) encrypts EMRFS objects in-transit between EMR cluster nodes & S3
    - **At-rest Encryption on Local disks:**
    - Open-source HDFS Encryption and Luks

- **EMR Cluster types:**
    - **Transient** (shut down when the job is done)
    - **Persistent** (continue to run after data processing job is complete)

- EMR is not about data ingestion is about massing analytic processing but it could take the data from Kinesis

- **Use Hadoop (Java) Apache**

- **Always uses HTTPS to send data between Amazon S3 and Amazon EC2, users also may encrypt the input data before upload it to S3 (uses Ipsec, Internet Protocol Security)**

- It is a service that processes vast amounts of data easily, from server logs stored on Amazon or from player data stored in a Dynamodb table.. for example

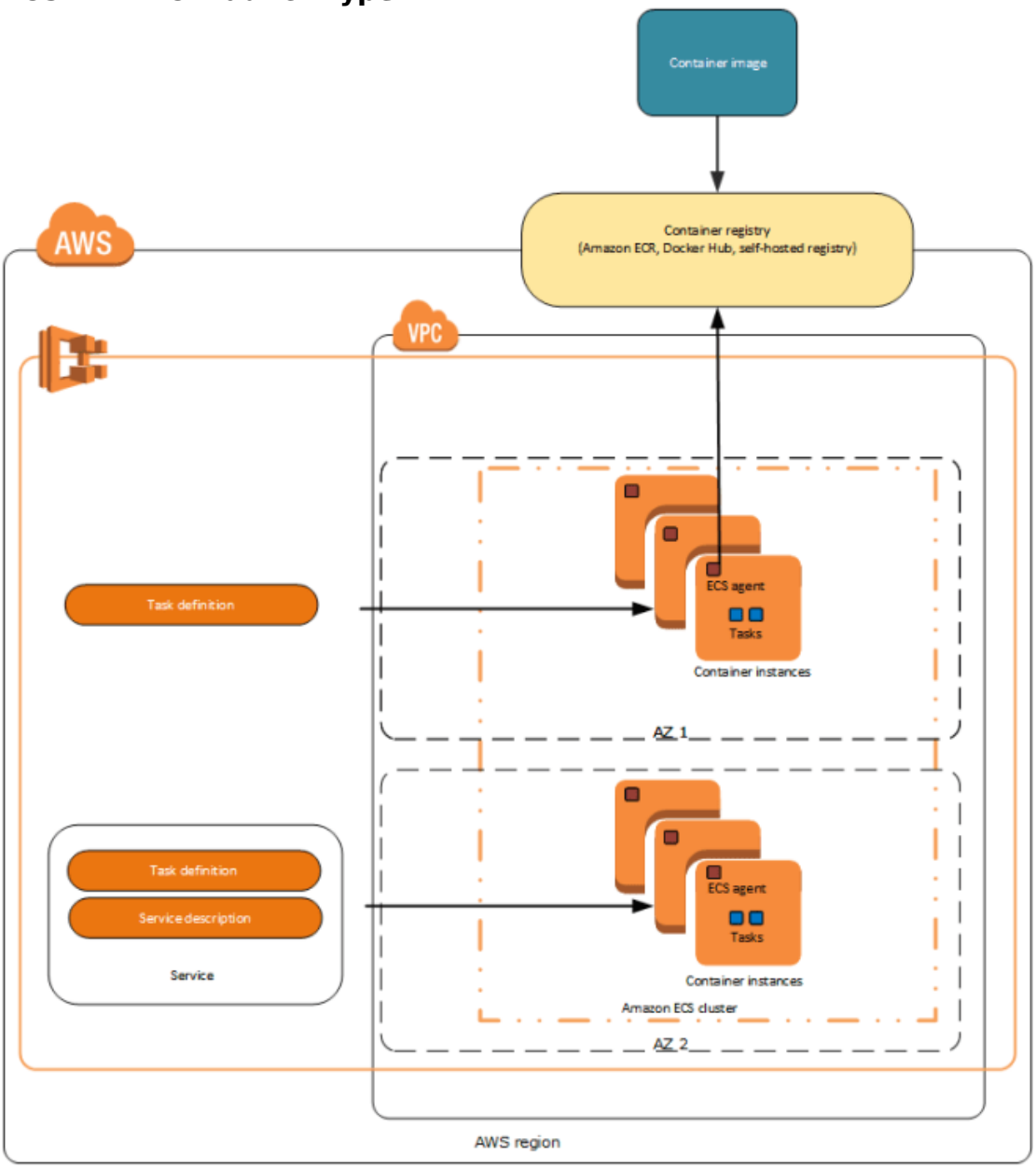
- **Amazon EMR cluster states:**
    - bootstrapping
    - running
    - waiting
    - terminated
    - terminate with errors

AWS EC2 Container Service ECS

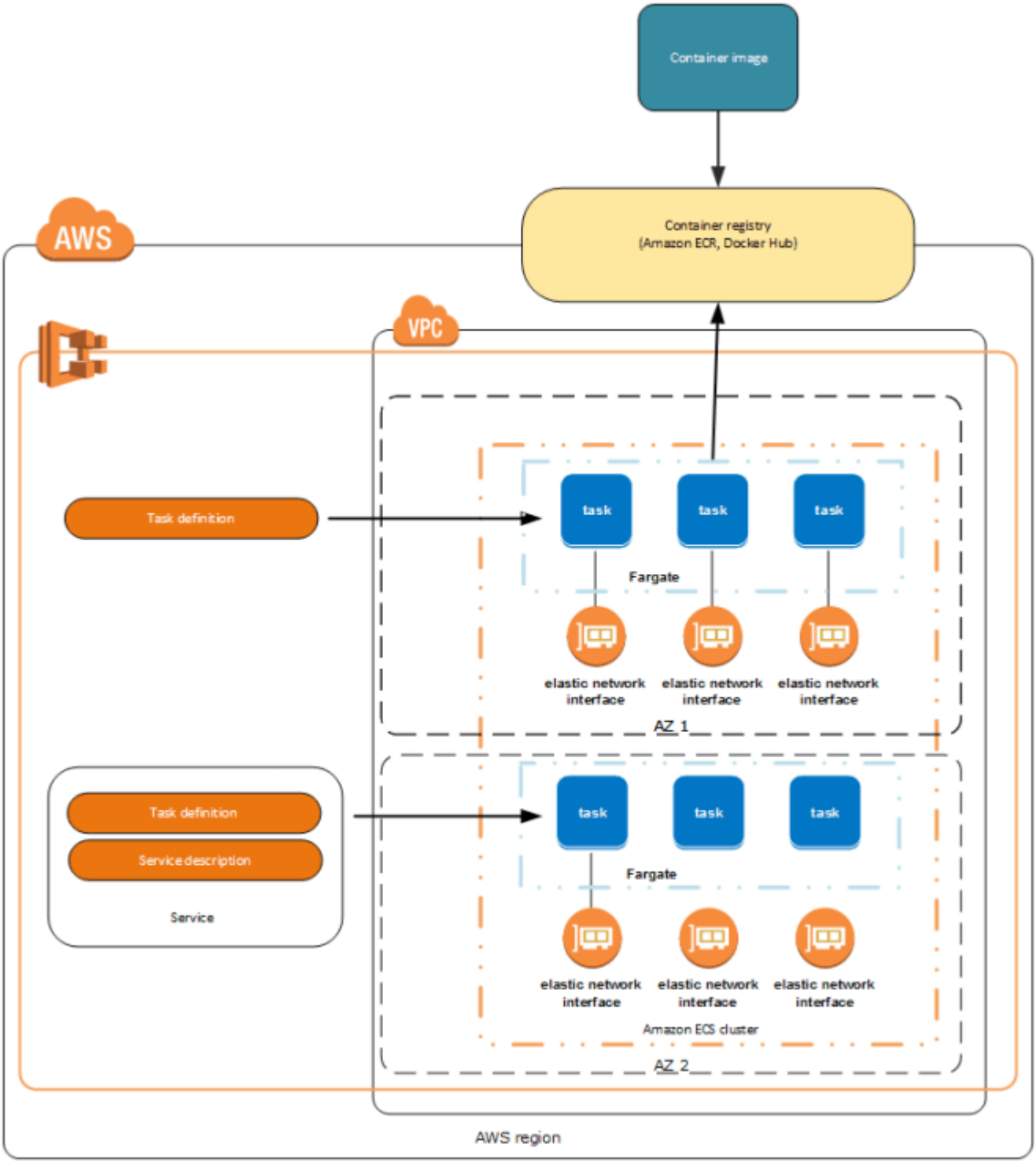
- AWS EC2 Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows running applications on a managed cluster of EC2 instances
- ECS eliminates the need to install, operate, and scale the cluster management infrastructure.
- ECS is a regional service that simplifies running application containers in a highly available manner across multiple AZs within a region
- ECS helps schedule the placement of containers across the cluster based on the resource needs and availability requirements.
- ECS allows integration of your own custom scheduler or third-party schedulers to meet business or application specific requirements.

ECS Elements

ECS with EC2 launch type



ECS with AWS Fargate



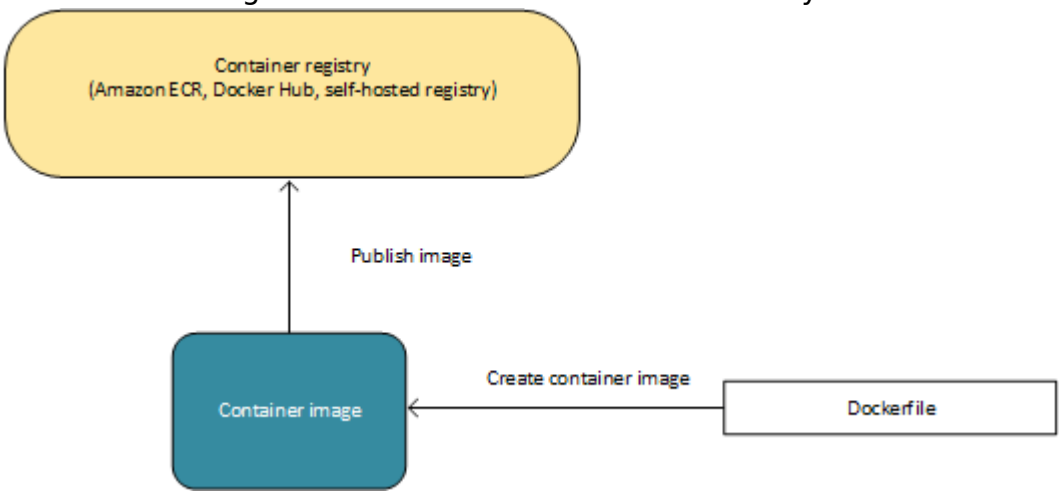
Containers and Images

- Applications deployed on ECS must be architected to run in docker containers, which is a standardized unit of software development, containing everything that the software application needs to run: code, runtime, system tools, system libraries, etc.
- Containers are created from a read-only template called an image.
- Images are typically built from a Dockerfile, and stored in a registry from which they can be downloaded and run on your container instances.

- ECS can be configured to access a private Docker image registry within a VPC, Docker Hub or is integrated with EC2 Container Registry (ECR)

Clusters

- Cluster is a logical grouping of EC2 container instances to run tasks using ECS
- ECS downloads the container images from the specified registry, and runs those images on the container instances within your cluster.

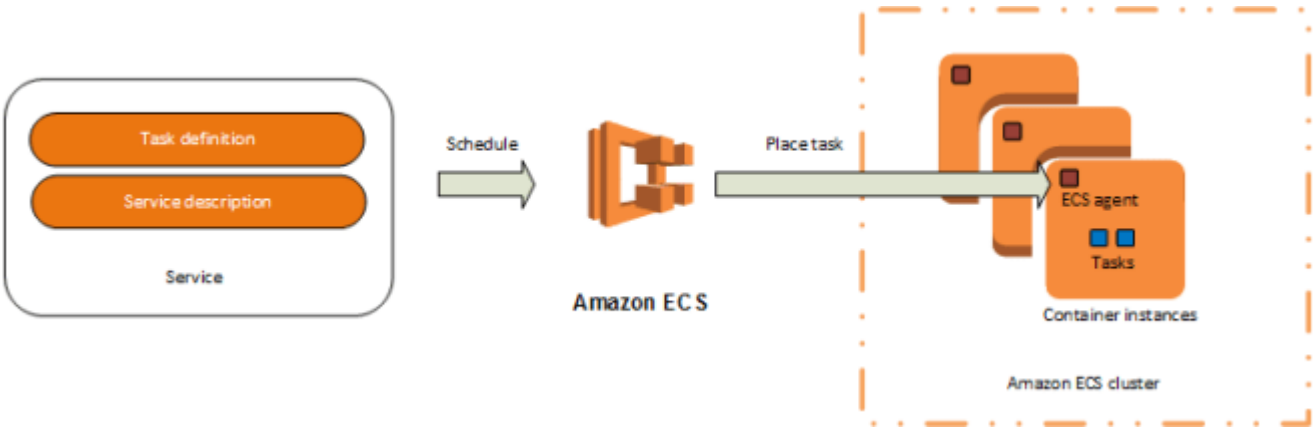


Task Definitions

- Task definition is a description of an application that contains one or more docker containers
- Task definition is needed to prepare application to run on ECS
- Task definition is a text file in JSON format that describes one or more containers that form your application.
- Task definitions specify various parameters for the application, such as containers to use, their repositories, ports to be opened, and data volumes

Tasks and Scheduling

- A task is the instantiation of a task definition on a container instance within the cluster.
- After a task definition is created for the application within ECS, you can specify the number of tasks that will run on the cluster.
- ECS task scheduler is responsible for placing tasks on container instances, with several different scheduling options available

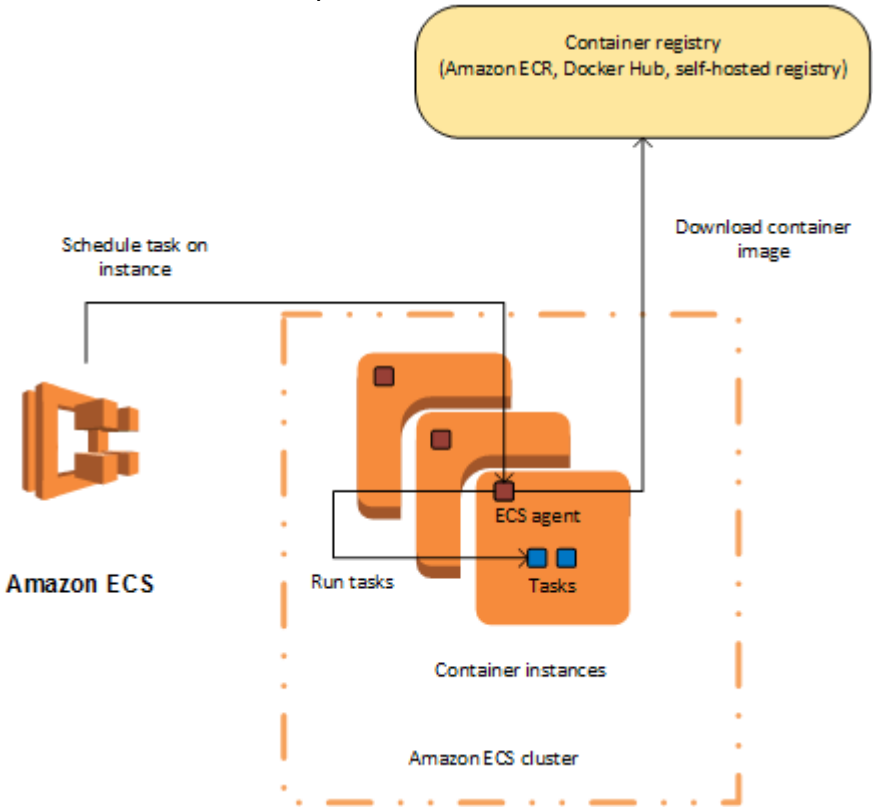


ECS Service

- ECS Service helps to run and maintain a specified number of instances of a task definition simultaneously.

Container Agent

- Container agent runs on each instance within an ECS cluster
- Container Agent sends information about the instance's current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS



ECS vs Elastic Beanstalk

- ECS helps in having a more fine-grained control for custom application architectures.
- Elastic Beanstalk is ideal to leverage the benefits of containers but just want the simplicity of deploying applications from development to production by uploading a container image.
- Elastic Beanstalk is more of an application management platform that helps customers easily deploy and scale web applications and services.
- With Elastic Beanstalk, specify container images to be deployed, with the CPU & memory requirements, port mappings and container links.
- Elastic Beanstalk abstracts the finer details and automatically handles all the details such as provisioning an ECS cluster, balancing load, auto-scaling, monitoring, and placing the containers across the cluster.

ECS vs Lambda

- **EC2 Container Service is a highly scalable Docker container management service that allows running and managing distributed applications in Docker containers.**
- **AWS Lambda is an event-driven task compute service that runs code (Lambda functions) in response to “events” from event sources like SES, SNS, DynamoDB & Kinesis Streams, CloudWatch etc.**

AWS Directory Services

- AWS Directory Service is a managed service offering, providing directories that contain information about the organization, including users, groups, computers, and other resources

- AWS Directory Services provides multiple ways including
    - **Simple AD** as a standalone directory service (<= 5000 users)
    - **AD Connecto**r to use On-Premise Microsoft Active Directory with other AWS services.
    - **AWS Directory Service** for **Microsoft Active Directory (Enterprise Edition**), also referred to as Microsoft AD (>=5000 users)

• Simple AD

- is a Microsoft Active Directory compatible directory from AWS Directory Service that is powered by Samba 4

- **is the least expensive option and the best choice if there are 5,000 or fewer users & don't need the more advanced Microsoft Active Directory features**

- supports commonly used Active Directory features such as user accounts, group memberships, domain-joining EC2 instances running Linux and Windows, kerberos-based single sign-on (SSO), and group policies

- **does not support features like DNS dynamic update, schema extensions, multi-factor authentication, communication over LDAPS, PowerShell AD cmdlets, and the transfer of FSMO roles**

- provides daily automated snapshots to enable point-in-time recovery

- **However, Trust relationships cannot be set-up between Simple AD and other Active Directory domains.**

• AD Connector

- Helps connect to an existing on-premises Active Directory to AWS

- is the best choice to leverage existing on-premises directory with AWS services

- is a proxy service for connecting on-premises Microsoft Active Directory to AWS without requiring complex directory synchronization technologies or the cost and complexity of hosting a federation infrastructure

- forwards sign-in requests to the Active Directory domain controllers for authentication and provides the ability for applications to query the directory for data

- enables consistent enforcement of existing security policies, such as password expiration, password history, and account lockouts, whether users are accessing resources on premises or in the AWS cloud

• Microsoft Active Directory (Enterprise Edition)

- is a feature-rich managed Microsoft Active Directory hosted on the AWS

- is the best choice if there are more than 5,000 users and need a trust relationship set up between an AWS hosted directory and on-premises directories.

- provides much of the functionality offered by Microsoft Active Directory plus integration with AWS applications

Microsoft AD connectivity options

- If the VGW used to connect to the On-Premise AD is not stable or has connectivity issues, the following options can be explored
  - **Simple AD**
    - least expensive option
    - provides an standalone instance for the Microsoft AD in AWS
    - No single point of Authentication or Authorization, as a separate copy is maintained
    - trust relationships cannot be set-up between Simple AD and other Active Directory domains
  - **Read-only Domain Controllers (RODCs)**
    - **works out as a Read-only Active Directory**
    - holds a copy of the Active Directory Domain Service (AD DS) database and respond to authentication requests
    - are typically deployed in locations where physical security cannot be guaranteed
    - they cannot be written to by applications or other servers.
    - helps maintain a single point to authentication & authorization controls, however needs to be synced
  - **Writable Domain Controllers**
    - are expensive to set-up
    - operate in a multi-master model; changes can be made on any writeable server in the forest, and those changes are replicated to servers throughout the entire forest

Key Notes - AWS Directory Service

- **Simple AD**
  - Microsoft Active Directory-compatible directory from AWS Directory Service that is powered by Samba 4.
  - To be used when there are fewer than 5,000 users.
  - is the least expensive option and the best choice if there are 5,000 or fewer users & don’t need the more advanced Microsoft Active Directory features
  - does not support features like DNS dynamic update, schema extensions, multi-factor authentication, communication over LDAPS, PowerShell AD cmdlets, and the transfer of FSMO roles
  -
- **AD Connector**
  - Proxy service for connecting an on premises Microsoft Active Directory to the AWS cloud.
  - Forwards sign-in requests to the Active Directory domain controllers for authentication.
  - To be used when you want to use current on premise active directory to the AWS cloud.
  - enables consistent enforcement of existing security policies, such as password expiration, password history, and account lockouts, whether users are accessing resources on premises or in the AWS cloud

- AWS Directory Service - **Microsoft AD (Active Directory Enterprise Edition)**
  - Managed Microsoft Active Directory hosted on AWS.
  - Integrates AWS applications with Active Directory.
  - To be used when there are more than 5,000 users and need a trust relationship set up between an AWS hosted directory and on-premises directories.

    - **Microsoft AD connectivity options**
      - If the VGW used to connect to the On-Premise AD is not stable or has connectivity issues, the following options can be explored
        - **Simple AD**
          - least expensive option
          - standalone instance for the Microsoft AD in AWS
          - No single point of Authentication or Authorization, as a separate copy is maintained
        - **Read-only Domain Controllers (RODCs)**
          - works out as a Read-only Active Directory
          - holds a copy of the Active Directory Domain Service (AD DS) database and respond to authentication requests.
          - helps maintain a single point to authentication & authorization controls, however needs to be synced
        - **Writable Domain Controllers**
          - are expensive to set-up
          - operate in a multi-master model;

AWS CloudFormation

- **AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provision and update them in an orderly**

- **CloudFormation consists of:**

    - **Template**
        - is an architectural diagram
        - a JSON or YAML-format, text-based file that describes all the AWS resources you need to deploy to run your application
        -
    - **Stack**
        - is the end result of that diagram, which is actually provisioned
        - is the set of AWS resources that are created and managed as a single unit when CloudFormation instantiates a template.

- **CloudFormation template can be used to set up the resources consistently and repeatedly over and over across multiple regions**

- Resources can be updated, deleted and modified in a controlled and predictable way, in effect applying version control to the infrastructure as done for software code

- **AWS CloudFormation Template consists of elements :-**
    - List of AWS resources and their configuration values
    - An optional template file format version number
    - An optional list of template parameters (input values supplied at stack creation time)
    - An optional list of output values like public IP address using the Fn::GetAtt function
    - An optional list of data tables used to lookup static configuration values for e.g., AMI names per AZ

- **CloudFormation supports Chef & Puppet Integration,** meaning that you can deploy and configure right down the application layer

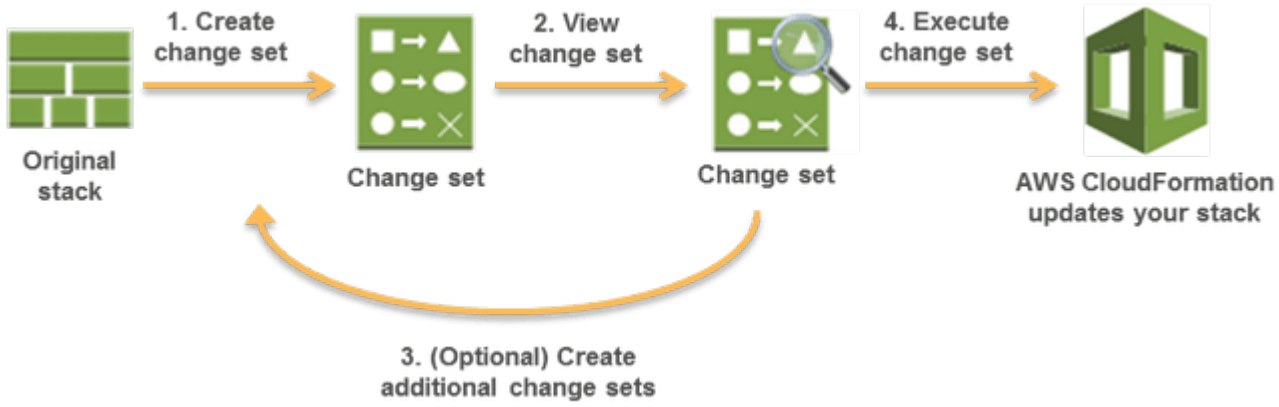
- CloudFormation provides a set of application bootstrapping scripts that enable you to install packages, files, and services on the EC2 instances by simply describing them in the CloudFormation template
- By default, automatic rollback on error feature is enabled, which will cause all the AWS resources that CloudFormation created successfully for a stack up to the point where an error occurred to be deleted.
- In case of automatic roll-back, charges would still be applied for the resources the time they were up and running

- CloudFormation provides a WaitCondition resource that acts as a barrier, blocking the creation of other resources until a completion signal is received from an external source e.g. application, or management system
- CloudFormation allows deletion policies to be defined for resources in the template for e.g. resources to be retained or snapshots can be created before deletion useful for preserving S3 buckets when the stack is deleted

Required Mainly for Developer, SysOps Associate & DevOps Professional Exam

AWS CloudFormation Concepts

- AWS CloudFormation, you work with templates and stacks:
    - Templates
        - act as blueprints for building AWS resources.
        - is a JSON or YAML formatted text file, saved with any extension, such as .json, .yaml, .template, or .txt.
        - have additional capabilities to build complex sets of resources and reuse those templates in multiple contexts for e.g. use input parameters to create generic and reusable templates
        - Name used for a resource within the template is a logical name but when CloudFormation creates the resource, it generates a physical name that is based on the combination of the logical name, the stack name, and a unique ID
    - Stacks
        - Stacks manage related resources as a single unit,
        - Collection of resources can be created, updated, and deleted by creating, updating, and deleting stacks.
        - All the resources in a stack are defined by the stack's AWS CloudFormation template
        - CloudFormation makes underlying service calls to AWS to provision and configure the resources in the stack and can perform only actions that the users have permission to do.
    - Change Sets
        - Change Sets presents a summary of the proposed changes CloudFormation will make when a stack is updated
        - Change Sets help check how the changes might impact running resources, especially critical resources, before implementing them
        - CloudFormation makes the changes to the stack only when the change set is executed, allowing you to decide whether to proceed with the proposed changes or explore other changes by creating another change set
        - Change sets don't indicate whether AWS CloudFormation will successfully update a stack for e.g. if account limits are hit or the user does not have permission



CloudFormation Access Control

- IAM
  - IAM can be applied with CloudFormation to access control for users whether they can view stack templates, create stacks, or delete stacks
  - IAM permissions need to be provided to the user to the AWS services and resources provisioned, when the stack is created
  - Before a stack is created, AWS CloudFormation validates the template to check for IAM resources that it might create

- Service Role
  - A service role is an AWS IAM role that allows AWS CloudFormation to make calls to resources in a stack on the user's behalf
  - By default, AWS CloudFormation uses a temporary session that it generates from the user credentials for stack operations.
  - For a service role, AWS CloudFormation uses the role's credentials.
  - When a service role is specified, AWS CloudFormation always uses that role for all operations that are performed on that stack.

Template Resource Attributes

- CreationPolicy Attribute
  - is invoked during the associated resource creation
  - can be associated with a resource to prevent its status from reaching create complete until AWS CloudFormation receives a specified number of success signals or the time-out period is exceeded
  - helps to wait on resource configuration actions before stack creation proceeds for e.g. software installation on an EC2 instance

- DeletionPolicy Attribute
  - preserve or (in some cases) backup a resource when its stack is deleted
  - By default, if a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource
  - To keep a resource when its stack is deleted,

        - specify Retain for that resource, to prevent deletion
        - specify Snapshot to create a snapshot before deleting the resource, if the snapshot capability is supported for e.g RDS, EC2 volume etc.

- DependsOn Attribute
    - helps specify that the creation of a specific resource follows another
    - resource is created only after the creation of the resource specified in the DependsOn attribute

- Metadata Attribute
    - enables association of structured data with a resource

- UpdatePolicy Attribute
    - defines AWS CloudFormation handles updates to the AWS::AutoScaling::AutoScalingGroup resource

Key notes - AWS CloudFormation

- **CloudFormation:**
    - Helps you model/set up AWS resources fast by creating a JSON template and take care of provisioning and configuring the resources indicated on the template {template}
    - CloudFormation template can be used to set up the resources consistently and repeatedly over and over across multiple regions
    - Resources can be updated, deleted and modified in and applying version control to the infrastructure as done for software code
    - CloudFormation supports Chef & Puppet Integration
    - By default, automatic **rollback** on error feature is enabled, which will cause all the AWS resources that CloudFormation created successfully for a stack up to the point where an error occurred to be deleted.
        - In case of automatic roll-back, charges would still be applied for the resources the time they were up and running
    - **CloudFormation consists of:**
        - **Template**
            - a JSON or YAML-format, text-based file that describes all the AWS resources you need to deploy to run your application
        - **Stack**
            - is the set of AWS resources that are created and managed as a single unit when CloudFormation instantiates a template.
    - **Change Sets**
        - Change Sets presents a summary of the proposed changes CloudFormation will make when a stack is updated
    - **CloudFormation Access Control**
        - **IAM**
            - IAM can be applied with CloudFormation to access control for users whether they can view stack templates, create stacks, or delete stacks | **IAM permissions need to be provided**
        - **Service Role**
            - A service role is an AWS IAM role that allows AWS CloudFormation to make calls to resources in a stack on the user’s behalf
    - Supports Amazon EC2 tagging.

AWS Config

- **AWS Config is a fully managed service that provides AWS resource inventory (audit), configuration history, and configuration change notifications to enable security and governance**

- It provides a detailed view of the configuration of AWS resources in the AWS account.

- It gives point-in-time and historical states and allows user to see changes visually in a time-line

- In cases where several configuration changes are made to a resource in quick succession (i.e., within a span of few minutes), AWS Config will only record the latest configuration of that resource; this represents the cumulative impact of that entire set of changes

- AWS Config does not cover all the AWS services and for the services unsupported the configuration management process can be automated using API and code and used to compare current and past data

AWS Config Use Case

- Security Analysis & Resource Administration
    - AWS Config enables continuous monitoring and governance over resource configurations and help evaluate them for any misconfiguration leading to security gaps or weakness

- Auditing & Compliance
    - AWS Config help maintain a complete inventory of all resources and their configurations attributes as well as point in time history
    - Ability to retrieve historical configurations can be very useful to ensure compliance with internal policies and best practices and for audits

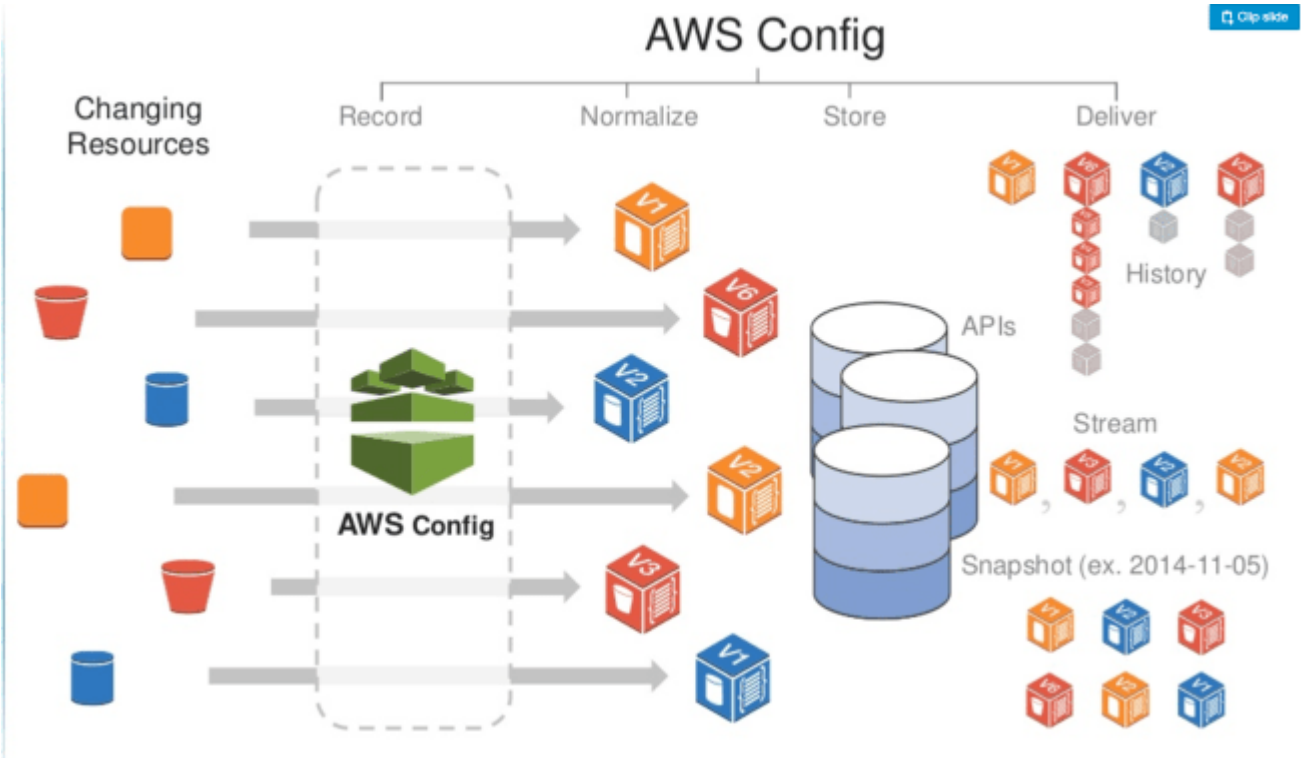
- Change Management
    - AWS Config helps understand relationships between resources so that the impact of the change can be pro-actively assessed
    - It can be configured to notify whenever resources are created, modified, or deleted without having to monitor these changes by polling the calls made to each resource

- Troubleshooting
    - AWS Config can help quickly identify and troubleshoot issues, by being able to use the historical configurations and compare the last working configuration to the one recent changed causing issues

- Discovery

- AWS Config help discover resources that exist within an account leading to better inventory and asset management
- Get a snapshot of the current configurations of the supported resources that are associated with the AWS account

AWS Config Concepts



- **AWS Resources**
  - AWS Resources are entities created and managed for e.g. EC2 instances, Security groups

- **AWS Config Rules**
  - AWS Config Rules helps define desired configuration settings for the resources or for the entire account
  - AWS Config continuously tracks the resource configuration changes against the rules and if violated marks the resource as non-compliant

- **Resource Relationship**
  - AWS Config discovers AWS resources in the account and then creates a map of relationships between AWS resources for e.g. EBS volume linked to an EC2 instance

- **Configuration Items**
  - A configuration item represents a point-in-time view of the supported AWS resource
  - Components of a configuration item include meta-data, attributes, relationships, current configuration, and related events.

- **Configuration Snapshot**
  - A configuration snapshot is a collection of the configuration items for the supported resources that exist in your account

- **Configuration History**
  - A configuration history is a collection of the configuration items for a given resource over any time period

- **Configuration Stream**
  - Configuration stream is an automatically updated list of all configuration items for the resources that AWS Config is recording

- **Configuration Recorder**
  - Configuration recorder stores the configurations of the supported resources in your account as configuration items
  - A configuration recorder needs to created and started for recording and by default records all supported services in the region

AWS Config Flow

- When AWS Config is turned on, it first discovers the supported AWS resources that exist in the account and generates a configuration item for each resource.

- AWS Config also generates configuration items when the configuration of a resource changes, and it maintains historical records of the configuration items of the resources from the time the configuration recorder is started.

- By default, AWS Config creates configuration items for every supported resource in the region, but can be customized to limited resource types.

- AWS Config keeps track of all changes to the resources by invoking the Describe or the List API call for each resource as well as related resources in the account

- Configuration items are delivered in a configuration stream to a S3 bucket

- AWS Config also tracks the configuration changes that were not initiated by the API. AWS Config examines the resource configurations periodically and generates configuration items for the configurations that have changed.

- **AWS Config rules, if configured,**
  - are evaluated continuously for resource configurations for desired settings.
  - Depending on the rule, resources are evaluated either in response to configuration changes or periodically.

- When AWS Config evaluates the resources, it invokes the rule's AWS Lambda function, which contains the evaluation logic for the rule.
- Function returns the compliance status of the evaluated resources.
- If a resource violates the conditions of a rule, the resource and the rule are flagged as non-compliant and a notification sent to SNS topic

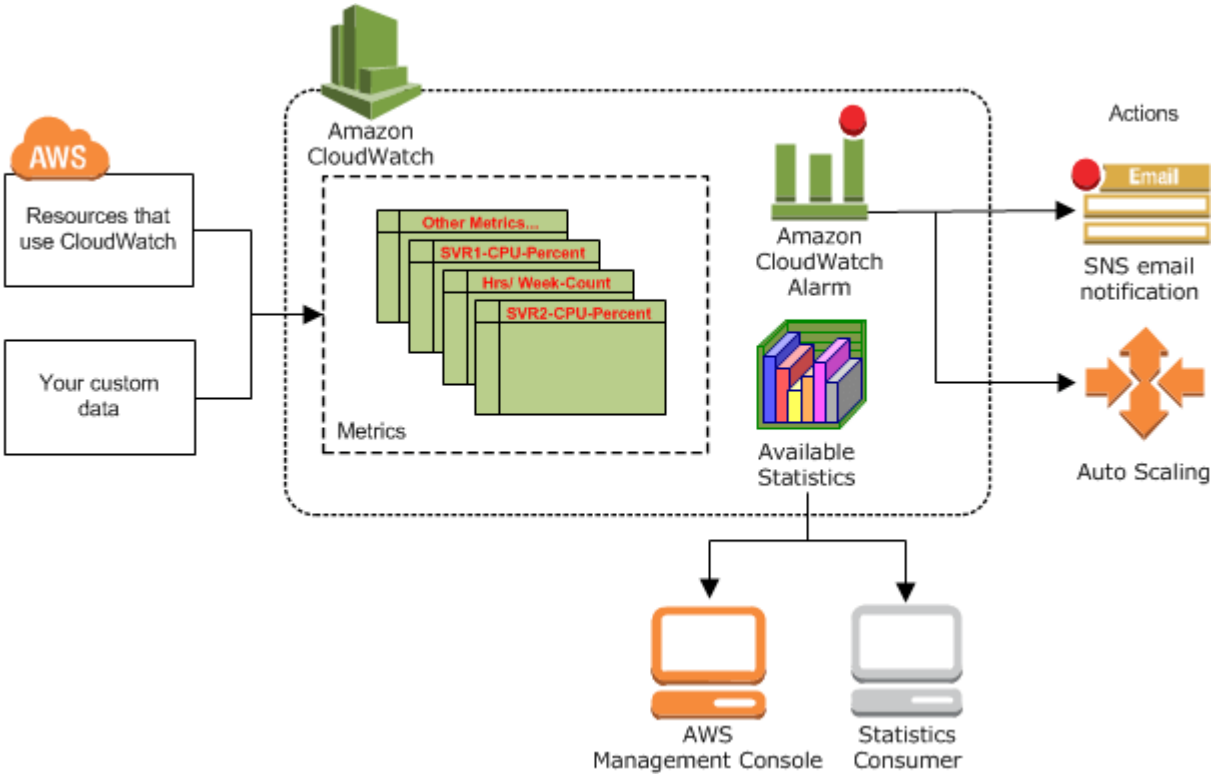
AWS Config – Key notes

- AWS Config Rules allow you to ensure that your environment is always compliant with the best practices. It continuously reviews your environment and alerts you regarding any gaps in the environment. For example, it can send a notification when an IAM user opens up port 22 (SSH default port) to the whole world. For this specific example, there is even a default AWS Managed Rule (i.e., "restricted-ssh"). .
- **AWS Fully managed service that provides:**
  - inventory
  - configuration history,
  - configuration change notifications to enable security and governance
- **AWS Config Use Case**
  - Security Analysis & Resource Administration
  - Auditing & Compliance
  - Troubleshooting
  - Discovery
- **AWS Config Flow**
  - maintains historical records of the configuration
  - When AWS Config is turned on, it first discovers the supported AWS resources that exist in the account and generates a configuration item for each resource.
  - keeps track of all changes to the resources by invoking the Describe or the List API call

AWS CloudWatch

- AWS CloudWatch monitors AWS resources and applications in real-time.
- CloudWatch can be used to collect and track metrics, which are the variables to be measured for resources and applications.
- **CloudWatch alarms can be configured**
  - **to send notifications or**
  - **to automatically make changes to the resources based on defined rules**
- In addition to monitoring the built-in metrics that come with AWS, custom metrics can also be monitored
- CloudWatch provides system-wide visibility into resource utilization, application performance, and operational health.
- By default, CloudWatch stores the log data indefinitely, and the retention can be changed for each log group at any time
- CloudWatch Alarm history is stored for only 14 days

CloudWatch Architecture



- CloudWatch collects various metrics from various resources
- These metrics, as statistics, are available to the user through Console, CLI

- CloudWatch allows creation of alarms with defined rules
    - to perform actions to auto scaling or stop, start, or terminate instances
    - to send notifications using SNS actions on your behalf

CloudWatch Concepts

Metrics

- Metric is the fundamental concept in CloudWatch.
- Uniquely defined by a name, a namespace, and one or more dimensions
- Represents a time-ordered set of data points published to CloudWatch.
- Each data point has a time stamp, and (optionally) a unit of measure
- Data points can be either custom metrics or metrics from other services in AWS.
- Statistics can be retrieved about those data points as an ordered set of time-series data that occur within a specified time window.
- When the statistics are requested, the returned data stream is identified by namespace, metric name, dimension, and (optionally) the unit.
- Metrics exist only in the region in which they are created
- CloudWatch stores the metric data for two weeks
- Metrics cannot be deleted, but they automatically expire in 14 days if no new data is published to them.
- NOTE: From
    - One minute data points are available for 15 days.
    - Five minute data points are available for 63 days.
    - One hour data points are available for 455 days (15 months).

Namespaces

- CloudWatch namespaces are containers for metrics.
- Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.
- AWS namespaces all follow the convention AWS/<service>, for e.g. AWS/EC2 and AWS/ELB
- Namespace names must be fewer than 256 characters in length.
- There is no default namespace. Each data element put into CloudWatch must specify a namespace

Dimensions

- A dimension is a name/value pair that uniquely identifies a metric.
- Every metric has specific characteristics that describe it, and you can think of dimensions as categories for those characteristics.
- Dimensions helps design a structure for the statistics plan.
- Dimensions are part of the unique identifier for a metric, whenever a unique name pair is added to one of the metrics, a new metric is created
- Dimensions can be used to filter result sets that CloudWatch query returns
- A metric can be assigned up to ten dimensions to a metric.

Time Stamps

- Each metric data point must be marked with a time stamp to identify the data point on a time series
- Time stamp can be up to two weeks in the past and up to two hours into the future.
- If no time stamp is provided, CloudWatch creates a time stamp based on the time the data element was received.
- All times reflect the UTC time zone when statistics are retrieved

Units

- Units represent the statistic's unit of measure for e.g. count, bytes, % etc

Statistics

- Statistics are metric data aggregations over specified periods of time
- Aggregations are made using the namespace, metric name, dimensions, and the data point unit of measure, within the specified time period

Periods

- Period is the length of time associated with a specific statistic.
- Each statistic represents an aggregation of the metrics data collected for a specified period of time.
- Although periods are expressed in seconds, the minimum granularity for a period is one minute.

Aggregation

- CloudWatch aggregates statistics according to the period length specified in calls to GetMetricStatistics.
- Multiple data points can be published with the same or similar time stamps. CloudWatch aggregates them by period length when the statistics about those data points are requested.
- Aggregated statistics are only available when using detailed monitoring.
- Instances that use basic monitoring are not included in the aggregates
- CloudWatch does not aggregate data across regions.

Alarms

- Alarms can automatically initiate actions on behalf of the user, based on specified parameters
- Alarm watches a single metric over a specified time period, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
- Alarms invoke actions for sustained state changes only i.e. the state must have changed and been maintained for a specified number of periods
- Action can be a
    - SNS notification
    - Auto Scaling policies
    - EC2 action – stop or terminate EC2 instances
- After an alarm invokes an action due to a change in state, its subsequent behavior depends on the type of action associated with the alarm.
    - For Auto Scaling policy notifications, the alarm continues to invoke the action for every period that the alarm remains in the new state.
    - For SNS notifications, no additional actions are invoked.

- An alarm has three possible states:
    - OK—The metric is within the defined threshold
    - ALARM—The metric is outside of the defined threshold
    - INSUFFICIENT_DATA—Alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state
- Alarms exist only in the region in which they are created.
- Alarm actions must reside in the same region as the alarm
- Alarm history is available for the last 14 days.
- Alarm can be tested by setting it to any state using the SetAlarmState API (mon-set-alarm-state command). This temporary state change lasts only until the next alarm comparison occurs.
- Alarms can be disabled and enabled using the DisableAlarmActions and EnableAlarmActions APIs (mon-disable-alarm-actions and mon-enable-alarm-actions commands).

Regions

- CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions.
- Custom Metrics
- CloudWatch allows publishing custom metrics with put-metric-data CLI command (or its Query API equivalent PutMetricData)
- CloudWatch creates a new metric if put-metric-data is called with a new metric name, else it associates the data with the specified existing metric
- put-metric-data command can only publish one data point per call
- CloudWatch stores data about a metric as a series of data points and each data point has an associated time stamp
- Creating a new metric using the put-metric-data command, can take up to two minutes before statistics can be retrieved on the new metric using the get-metric-statistics command and can take up to fifteen minutes before the new metric appears in the list of metrics retrieved using the list-metrics command.

- **CloudWatch allows publishing**
    - **Single data point**
        - Data points can be published with time stamps as granular as one-thousandth of a second, CloudWatch aggregates the data to a minimum granularity of one minute
        - CloudWatch records the average (sum of all items divided by number of items) of the values received for every 1-minute period, as well as number of samples, maximum value, and minimum value for the same time period
        - CloudWatch uses one-minute boundaries when aggregating data points
    - Aggregated set of data points called a statistics set
        - Data can also be aggregated before being published to CloudWatch
        - Aggregating data minimizes the number of calls reducing it to a single call per minute with the statistic set of data
        - Statistics include Sum, Average, Minimum, Maximum, Data Sample

- If the application produces data that is more sporadic and have periods that have no associated data, either a the value zero (0) or no value at all can be published
- However, it can be helpful to publish zero instead of no value
    - to monitor the health of your application for e.g. alarm can be configured to notify if no metrics published every 5 minutes
    - to track the total number of data points
    - to have statistics such as minimum and average to include data points with the value 0.

AWS CloudWatch – Key notes

- **AWS CloudWatch monitors AWS resources and applications in real-time.**

- CloudWatch can be used to collect and track metrics, which are the variables to be measured for resources and applications.

- **CloudWatch alarms can be configured**

    - **to send notifications or**
    - **to automatically make changes to the resources based on defined rules**

- In addition to monitoring the built-in metrics that come with AWS, custom metrics can also be monitored

- **Alarms**
    - Alarms can automatically initiate actions on behalf of the user, based on specified parameters
    - Alarm watches a single metric over a specified time period, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
    - Alarms invoke actions for sustained state changes only i.e. the state must have changed and been maintained for a specified number of periods
    - **Action can be a**
        - **SNS notification**
        - **Auto Scaling policies**
        - **EC2 action – stop or terminate EC2 instances**
    - **After an alarm invokes an action due to a change in state**, its subsequent behavior depends on the type of action associated with the alarm:
        - **For Auto Scaling policy notifications**, the alarm continues to invoke the action for every period that the alarm remains in the new state.
        - **For SNS notifications**, no additional actions are invoked.

    - **An alarm has three possible states:**
        - **OK**—The metric is within the defined threshold

        - **ALARM**—The metric is outside of the defined threshold
        - **INSUFFICIENT_DATA**—Alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state

    - Alarms exist only in the region in which they are created and Alarm actions must reside in the same region as the alarm

    - Alarm history is available for the last 14 days.

    - Alarm can be tested by setting it to any state using the SetAlarmState API (mon-set-alarm-state command). This temporary state change lasts only until the next alarm comparison occurs.

    - Alarms can be disabled and enabled using the DisableAlarmActions and EnableAlarmActions APIs (mon-disable-alarm-actions and mon-enable-alarm-actions commands).

    - Graph and Zoom in to see data a shorter time period on the AWS console.

    - To set the threshold, set a target value and choose whether the alarm will trigger when the value is greater than > , greater than or equal to >= less than < or less than or equal to <= that value

    - With **cpucredit and cpuutilization** metrics, you can see the consumed credits

- **CloudWatch can be accessed using**

    - **AWS CloudWatch console**
    - **CloudWatch CLI**
    - **AWS CLI**
    - **CloudWatch API**
    - **AWS SDKs**

Identity & Access Management (IAM)

-

- **AWS Identity and Access Management (IAM)**
  - IAM is used to access AWS resources. It cannot be used to access OS of servers. Like other AWS resources, IAM falls under shared responsibility model. This means you are responsible for assigning the right access to the right user or resources, AWS is not.

  - **To secure control access to AWS resources for your users and keep your account credentials private.**

  - **Is s used to control:**
    - **Identity –**
      - who can use your AWS resources (authentication) and helps to provide authentication for people and processes in your AWS account

    - **Access**
      - what resources they can use and in what ways (authorization)

  - With IAM, multiple IAM users can be created under the AWS account or temporary access can be enabled through identity federation with corporate directory third party providers

  - IAM also enables access to resources across AWS accounts.

- **IAM Features**

  - **Shared access to your AWS account**
    - Grant other people permission to administer and use resources

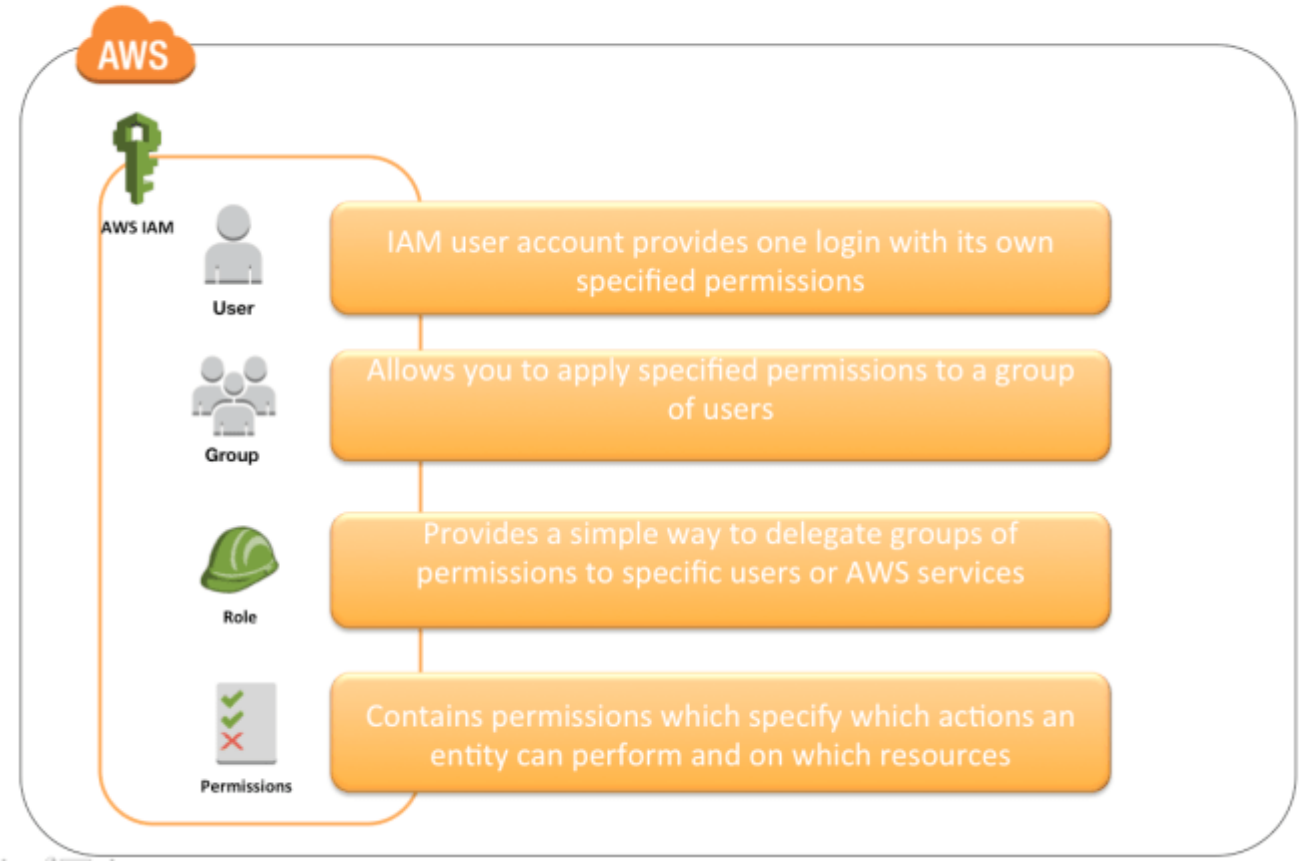
  - **Granular permissions**
    - Set granular permissions to users as required to perform their job

  - Secure access to AWS resources for applications that run on EC2 (temporary credentials)

  - **Identity federation**
    - IAM allows users to access AWS resources, without requiring the user to have accounts with AWS, by providing temporary credentials for e.g. through corporate network or Google or Amazon authentication
  - **Identity information for assurance**
    - CloudTrail can be used to receive log records that include information about those who made requests for resources in the account.
  - **PCI DSS Compliance**
    - IAM supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated

as being Payment Card Industry Data Security Standard (PCI DSS) compliant

AWS IAM Identities



- **Account Root User**
  - **Root Account Credentials are the email address and password** with which you sign-in into the AWS account
  - IAM Best Practice – Do not use or share the Root account once the AWS account is created, instead create a separate user with admin privilege
  - **An Admin account has full access except the accounts security credentials, billing information and ability to change password**

- **IAM Users**
  - IAM user represents the person or service who uses the access to interact with AWS.
  - IAM Best Practice – Create Individual Users and grant least Privilege as possible
  - User credentials can consist of the following
  - Password to access AWS services through AWS Management Console
  - Access Key/Secret Access Key to access AWS services through API, CLI or SDK
  - IAM user starts with no permissions

Credential Type	Use	Description
Passwords	AWS root account or IAM user account login to the AWS Management Console	A string of characters used to log into your AWS account or IAM account. AWS passwords must be a minimum of 6 characters and may be up to 128 characters.
Multi-Factor Authentication (MFA)	AWS root account or IAM user account login to the AWS Management Console	A six-digit single-use code that is required in addition to your password to log in to your AWS Account or IAM user account.
Access Keys	Digitally signed requests to AWS APIs (using the AWS SDK, CLI, or REST/Query APIs)	Includes an access key ID and a secret access key. You use access keys to digitally sign programmatic requests that you make to AWS.
Key Pairs	• SSH login to EC2 instances • CloudFront signed URLs	A key pair is required to connect to an EC2 instance launched from a public AMI. The keys that Amazon EC2 uses are 1024-bit SSH-2 RSA keys. You can have a key pair generated automatically for you when you launch the instance or you can upload your own.
X.509 Certificates	• Digitally signed SOAP requests to AWS APIs • SSL server certificates for HTTPS	X.509 certificates are only used to sign SOAP-based requests (currently used only with Amazon S3). You can have AWS create an X.509 certificate and private key that you can download, or you can upload your own certificate by using the Security Credentials page.

- **IAM Groups**
  - IAM Best Practice – Use groups to assign permissions to a collection of IAM Users sharing the same job function
  - **A group can have multiple users, while a user can belong to multiple groups (10 max)**
  - AWS does not provide any default group
  - Deletion of the groups requires you to detach users and managed policies and delete any in-line policies before deleting the group. With AWS management console, the deletion and detachment is taken care of.
  -
- **MultiFactor Authentication (MFA)**
  - For increased security and to help protect the AWS resources
  - IAM Best Practice – Enable MFA on Root accounts and privilege users
  - Multi-Factor Authentication can be configured using:
    - **Security token-based**
      - AWS Root user or IAM user can be assigned a hardware/virtual MFA device
    - **SMS text message-based (Preview Mode) – Only for users**
      - IAM user can be configured with the phone number
  - MFA needs to enabled on the Root user and IAM user separately
  - **If the MFA device stops working or is lost, you won't be able to login into the AWS console and would need to reach out to AWS support to deactivate MFA**

IAM Access Management

- **IAM Access Management is all about Permissions and Policies**
- **Permission allows you to define who has access and what actions can they perform**
- **IAM Policy helps to fine tune the permissions granted to the policy owner**
- IAM Policy is a document that formally states one or more permissions.
- Most restrictive Policy always wins
- IAM Policy is defined in the **JSON (**JavaScript Object Notation) format
- IAM policy basically states “Principal A is allowed or denied (effect) to perform Action B on Resource C given Conditions D are satisfied”
- **For example:**
- {
  “Version”: “2012-10-17",
  “Statement”: {
  “Principal“: {“AWS”: [“arn:aws:iam::ACCOUNT-ID-WITHOUT-HYPHENS:root”]},
  “Action“: “s3:ListBucket”,
  “Effect“: “Allow”,
  “Resource“: “arn:aws:s3:::example_bucket”,
  “Condition“: {“StringLike”: {
  “s3:prefix”: [“home/${aws:username}/”]
  }
  }
  }
  }

- Identity-based, or IAM permissions
  - Identity-based, or IAM permissions are attached to an IAM user, group, or role and specify what the user, group or role can do

- Resource-based permissions
  - Resource-based permissions are attached to a resource for e.g. S3, SNS and specifies both who has access to the resource (Principal) and what actions they can perform on it (Actions)

- **Managed Policies and Inline Policies**
  - Managed policies
    - Managed policies are Standalone policies that can be attached to multiple users, groups, and roles in an AWS account and apply only to identities (users, groups, and roles) but not to resources.
    - **Two types of managed policies:**
      - **AWS managed policies**
        - Managed policies that are created and managed by AWS.
      - **Customer managed policies**
        - Managed policies are standalone and custom policies created and administered by you.

- In-line policies
  - In-line policies are created and managed by you, and are embedded directly into a single user, group, or role.
  - Deletion of the Entity (User, Group or Role) or Resource deletes the In-Line policy as well

- **IAM Policy Simulator**
  - **IAM Policy Simulator helps test and troubleshoot IAM resource-based policies in the following ways :-**
    - Test IAM based policies and Resource based policies.
    - Test the policies with selected services, actions, and resources
    - Simulate real-world scenarios
    - Identify which specific statement in a policy results in allowing or denying access to a particular resource or action.

- **Credential Report (to audit such as password and access key rotation.)**
  - **IAM allows you to generate and download a credential report that lists all users in the account and the status of their various credentials, including passwords, access keys, and MFA devices.**
  - IAM Best Practice – Perform Audits and Remove all unused users and credentials

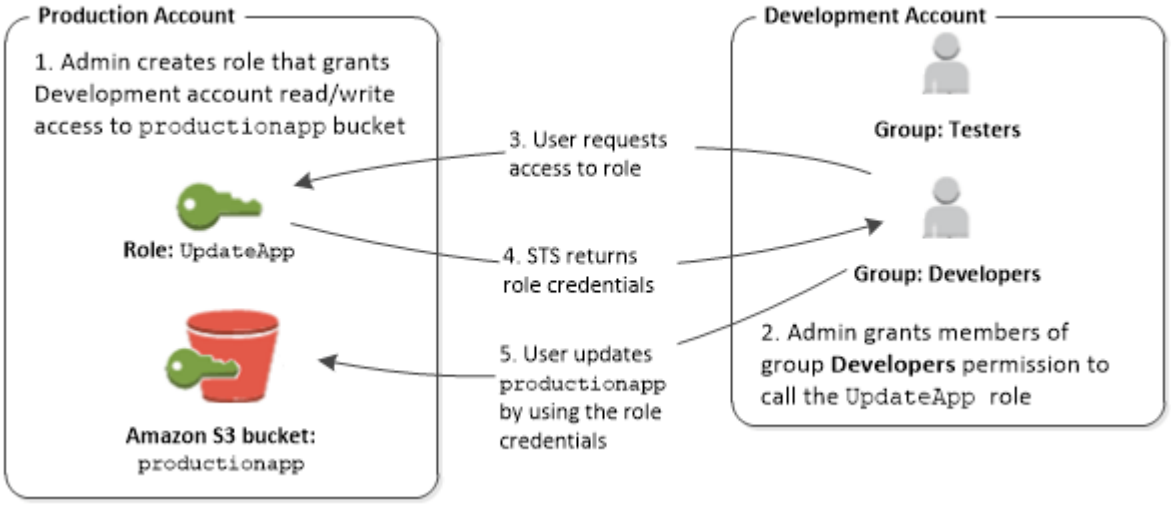
- **AWS IAM Role**

  - **IAM Role plays a very important role in the following scenarios:**
    - Interact with other AWS services (like EC2 or EMR), for example, A EC2 running an application that needs to access other AWS services
    - Allowing users from different AWS accounts have access to AWS resources in different account, instead of having to create users
    - Company uses a Corporate Authentication mechanism and don't want the User to authenticate twice or create duplicate users in AWS
    - Applications allowing login through external authentication mechanism e.g. Amazon, Facebook, Google etc
    - **Role involves defining two policies**
      - **Trust policy**
        - Trust policy defines – who can assume the role
      - **Permissions policy**
        - Permissions policy defines – what they can access
- **Federation is creating a trust relationship between an external Identity Provider (IdP) and AWS**
  - Users can also sign in to an enterprise identity system that is compatible with SAML
  - Users can sign in to a web identity provider, such as Login with Amazon, Facebook, Google, or any IdP that is compatible with OpenID connect (OIDC).
  - When using OIDC and SAML 2.0 to configure a trust relationship between these external identity providers and AWS, the user is

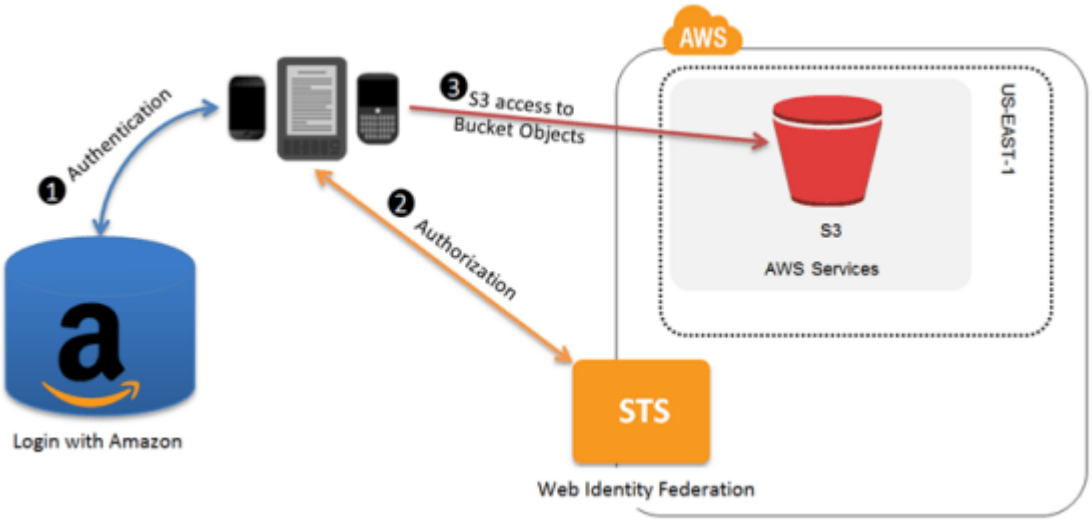
assigned to an IAM role and receives temporary credentials that enables the user to access AWS resources

- IAM Best Practices:
    - Use roles for applications running on EC2 instances
    - Delegate using roles instead of sharing credentials

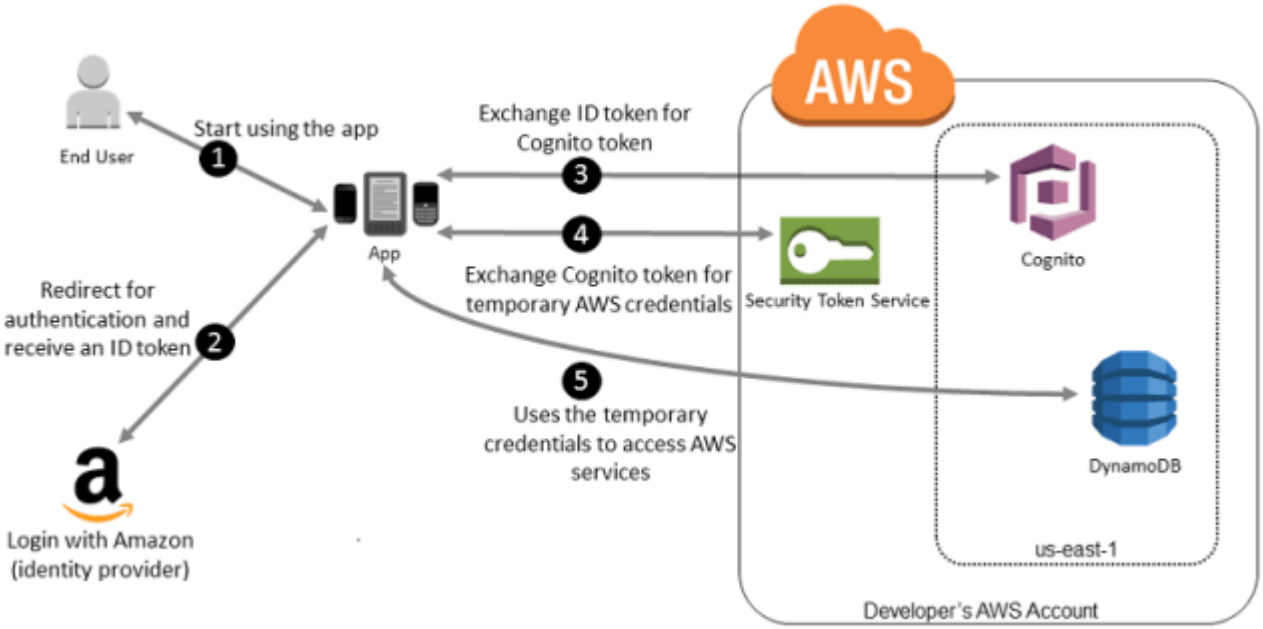
- **AWS STS & Temporary Credentials**
    - AWS Security Token Service (STS) helps create and provide trusted users with temporary security credentials that can control access to AWS resources
    - **The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).**
    - **AWS STS is a global service with a single endpoint**
        - **Endpoints:** The AWS Security Token Service (STS) has a default endpoint of https://sts.amazonaws.com that maps to the US East (N. Virginia) region. Additional regions are available and are activated by default.
    - **Recording API requests:** STS supports AWS CloudTrail, which is a service that records AWS calls for your AWS account and delivers log files to an Amazon S3 bucket. By using information collected by CloudTrail, you can determine what requests were successfully made to STS, who made the request, when it was made, and so on.
-
- **Cross-Account access Roles**
    - IAM users can be granted permission to switch roles within the same AWS account or to roles defined in other AWS accounts that you own.
    - Roles can also be used to delegate permissions to IAM users from AWS accounts owned by Third parties
    - You must explicitly grant the users permission to assume the role.
        - Users must actively switch to the role using the AWS Management Console.
        - Multi-factor authentication (MFA) protection can be enabled for the role so that only users who sign in with an MFA device can assume the role

- 

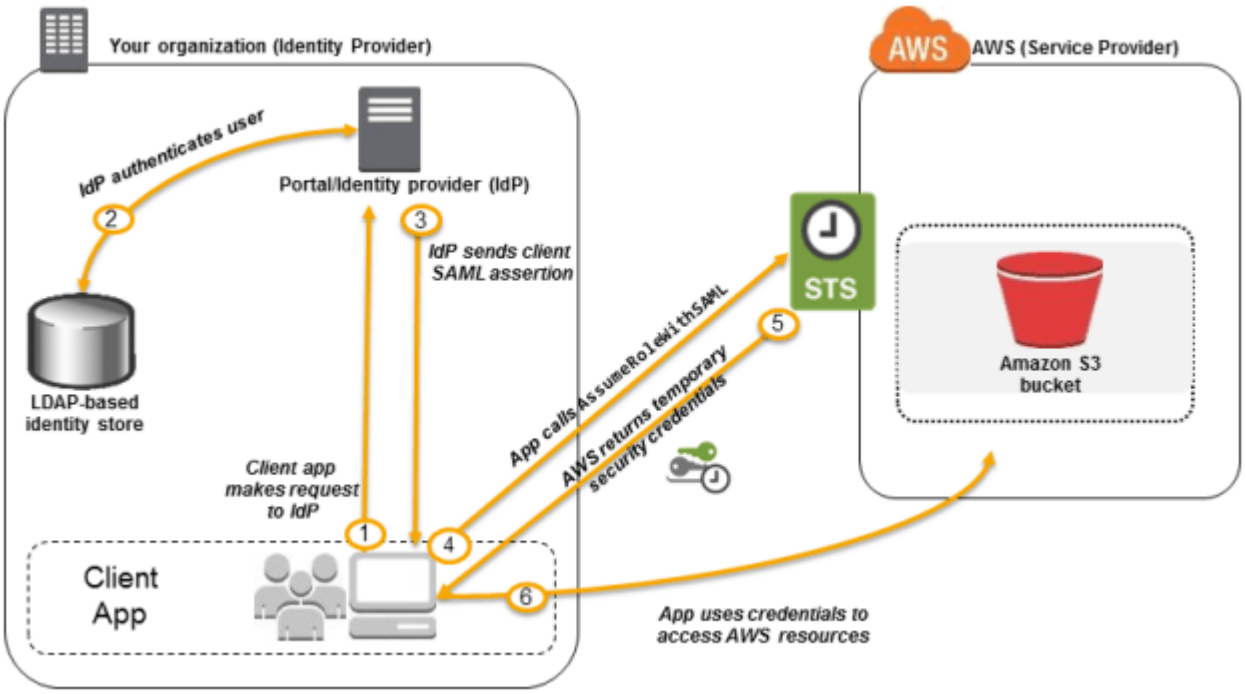
- **IAM Role - Identity Providers and Federation**
  - **Identity Provider can be used to grant external user identities permissions to AWS resources** without having to be created within your AWS account.
  - External user identities can be authenticated either through the organization's authentication system or through a well-know identity provider such as login with Amazon, Google etc.
  - To use an IdP, you create an IAM identity provider entity to establish a trust relationship between you AWS account and the IdP.
  - IAM supports IdPs that are compatible with OpenID Connect (OIDC) or SAML 2.0 (Security Assertion Markup Language 2.0)Web Identity Federation

- **Web Identity Federation - Complete Process Flow**
- 

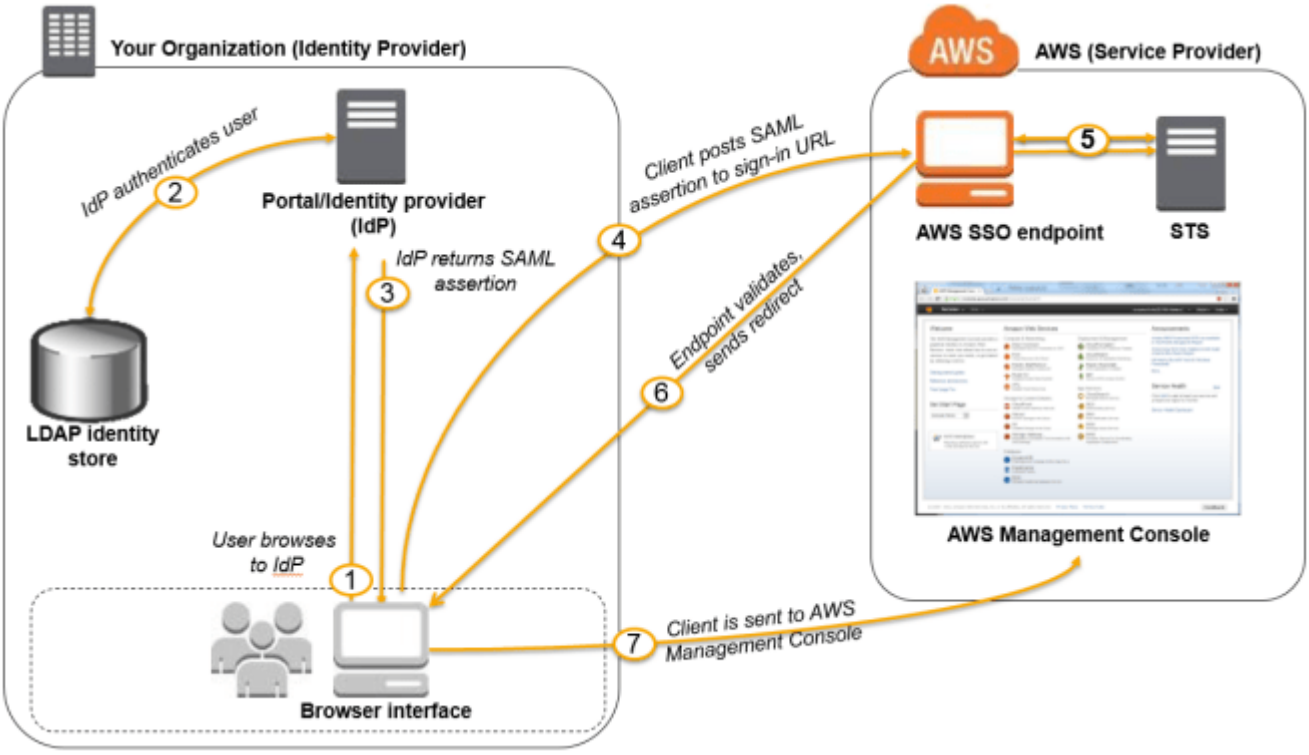
- **Web Identity Federation with Cognito**
  - Use Amazon Cognito as the identity broker for almost all web identity federation scenarios
  -



- **SAML 2.0-based Federation**
  - AWS supports identity federation with SAML 2.0 (Security Assertion Markup Language 2.0), an open standard that many identity providers (IdPs) use.
  - **SAML 2.0 based federation feature enables federated single sign-on (SSO)**, so users can log into the AWS Management Console or call the AWS APIs without having to create an IAM user for everyone in the organization
  - By using SAML, the process of configuring federation with AWS can be simplified by using the IdP's service instead of writing custom identity proxy code.
  - This is useful in organizations that have integrated their identity systems (such as Windows Active Directory or OpenLDAP) with software that can produce SAML assertions to provide information about user identity and permissions (such as Active Directory Federation Services or Shibboleth)



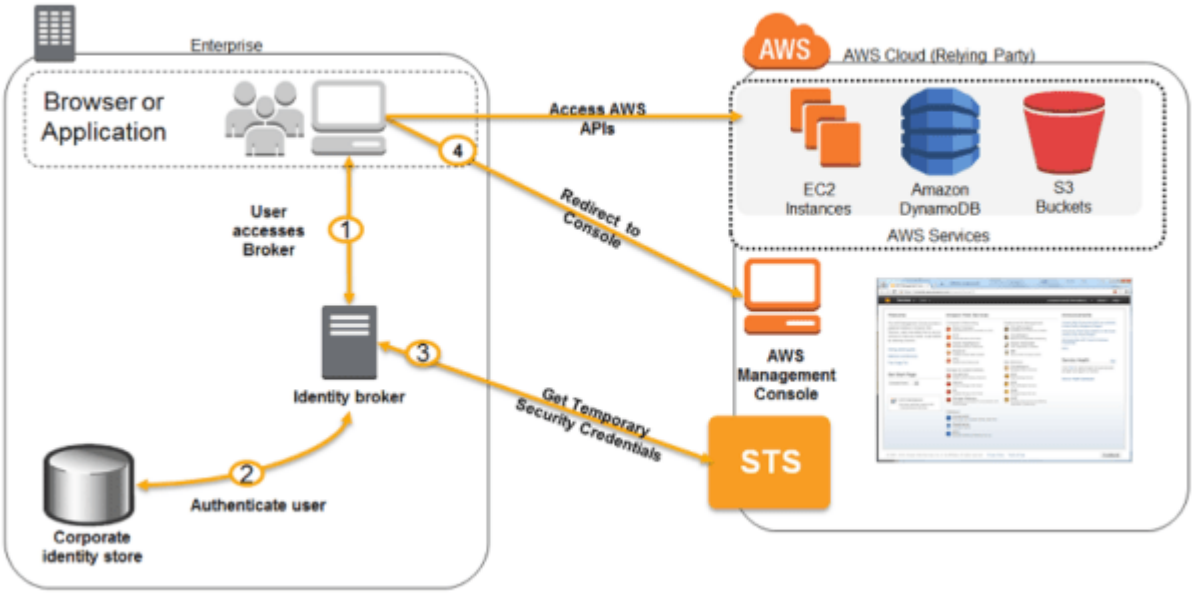
- SAML 2.0 based federation can also be used to grant access to the federated users to the AWS Management console. This requires the use of the AWS SSO endpoint instead of directly calling the AssumeRoleWithSAML API. The endpoint calls the API for the user and returns a URL that automatically redirects the user's browser to the AWS Management Console.



•

- **Custom Identity broker Federation**

Sample workflow using a custom identity broker application



- **If the Organization doesn’t support SAML compatible IdP, a Custom Identity Broker can be used to provide the access**
- **Custom Identity Broker should perform the following steps**
- **AWS IAM Roles vs Resource Based Policies**
    - AWS allows granting cross-account access to AWS resources, which can be done using IAM Roles or Resource Based policies
    - **IAM Roles**
        - **Roles can be created to act as a proxy to allow users or services to access resources**
        - **Roles supports trust policy** which helps determine who can **access** the **resources** and **permission policy** which helps to **determine what they can access**
        - Roles can be used to provision access to almost all the AWS resources
        - Permissions provided to the User through the Role can be further restricted per user by passing optional policy to the STS request.
            - **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).

- **Endpoints:** The AWS Security Token Service (STS) has a default endpoint of https://sts.amazonaws.com that maps to the US East (N. Virginia) region. Additional regions are available and are activated by default.

- **Recording API requests:** STS supports AWS CloudTrail, which is a service that records AWS calls for your AWS account and delivers log files to an Amazon S3 bucket. By using information collected by CloudTrail, you can determine what requests were successfully made to STS, who made the request, when it was made, and so on.

  - **Resource based Policies**
    - Resource based policy allows you to attach a policy directly to the resource that you want to share, instead of using a role as a proxy.
    - Resource-based policy specifies who can access that resource and what they can access
    - With Cross-account access with a resource-based policy, User still works in the trusted
    - User can work on the resources from both the accounts at the same time and this can be useful for scenarios for e.g. copying of objects from one bucket to the other
    - **Resources which support resource-based policies:**
      - Amazon S3 allows you to define Bucket policy to grant access to the bucket and the objects
      - Amazon Simple Notification Service (SNS)
      - Amazon Simple Queue Service (SQS)
      - Amazon Glacier Vaults
      - AWS OpsWorks stacks
      - AWS Lambda functions

- **AWS IAM Best Practices**

  - Root Account -Don’t use & Lock away access keys and Do not generate the access keys for it, if not required
  - Enable AWS multifactor authentication (MFA) on your AWS account
  - Start by creating a IAM User with Administrator role, which has access to all resources as the Root user except to the account’s security credentials
  - Create individual users and groups, and define the relevant permissions for each group as per the job function, and then associate IAM users to those groups.
  - **IAM user, by default, is created with no permissions**
  - Users should be granted LEAST PRIVILEGE as required to perform a task.

- Enforce user to create strong passwords and enforce them to rotate their passwords periodically
- Enable a strong password policy to define passwords
- **MFA – Enable MFA for privileged users**
- Use roles for applications running on EC2 instances instead of creating IAM user and hard-coding the credentials within that application.
- Allow users from same AWS account, another AWS account, or externally authenticated users (either through any corporate authentication service or through Google, Facebook etc) to use IAM roles to specify the permissions which can then be assumed by them
- Use the Credential report that lists all IAM users in the account and status of their various credentials, including passwords, access keys, and MFA devices and usage pattern to figure out what can be removed
    - **Passwords and access keys that have not been used recently might be good candidates for removal.**
- Define conditions under which IAM policies allow access to a resource.
- Conditions would help provide finer access control to the AWS services and resources for e.g. access limited to specific ip range or allowing only encrypted request for uploads to S3 buckets etc.
- Enable logging features provided through CloudTrail, S3, CloudFront in AWS to determine the actions users have taken in the account and the resources that were used.
- Log files show the time and date of actions, the source IP for an action, which actions failed due to inadequate permissions, and more.

- **Some tips:**

- You might be asked the authentication type for the below use cases in the exam:

Use Case	Solution
Operating System Access	Active Directory LDAP
	Machine – specific accounts
Application Access	Active Directory
	Amazon Cognito
AWS Resources (very important for AWS Users you need IAM)	IAM

-
- IAM is an entity which lets you manage AWS resources. It can be temporary or permanent.
-
- **There are three types of principles:**
- **Root Users**
    - The account which you create your AWS Account (email and password)
    - Best practice is not to use it for daily tasks and lock it away
    - It has access to all AWS resources.
- **IAM Users**
    - A user accounts
    - A user account is persistent, it will not expire. Unless, the administrator deletes it.
    - **A new IAM user, has no access key nor a password.**
- **Roles/Temporary Security Token**

    - Roles are granted access privileges for a set duration of time. (compared to USER which is persistent roles are not)
    - When a service tries to access a resource, AWS provides a Temporary Security Token. It is valid from 15 minutes to 36 hours.
    - Roles do not have user name or access keys.
    - Use cases of Roles:
        - EC2-Server: Instead of saving S3 credentials in EC2 server, a role can be attached to the EC2 server to access S3 without user-name and password.
        - Granting permission to other AWS Account to access your resources.
- **Federation:**
    - Let’s roles to be attached to non- AWS IAM Users. Example: A company that has migrated its resources to AWS, does not need to create the same users in AWS. It can integrate IAM with its Active Directory and attach roles to the users in the Active Directory.
    - IAM can integrate with the below providers:
        - OpenID for (Cognito) Google, Facebook auth.
        - SAML for Active Directory
- **IAM policy**
    - Set of action that can be executed on resources.
- ```
  {
    "Version": "2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::example_bucket"
    }
  }
  ```
    - It is written in JSON and can be attached to a group, user or role.
    - **Effects**: Can be either Allow or Deny.
    - **Action**: Which AWS service does this policy apply
    - **Resource**: The Amazon Resource Name (ARN).
- **Condition:** List of conditions that is required for the action to be allowed.
-
- Associating a policy to an IAM user can be attached in the following methods:
    - Inline Policy.
    - Managed Policy.
    - Group Policy.
  ```

AWS IAM Roles vs Resource Based Policies

AWS allows granting cross-account access to AWS resources, which can be done using IAM Roles or Resource Based policies

IAM Roles

- **Roles can be created to act as a proxy to allow users or services to access resources**
- Roles supports trust policy which helps determine who can access the resources and permission policy which helps to determine what they can access
- User who assumes a role temporarily gives up his or her own permissions and instead takes on the permissions of the role. When the user exits, or stops using the role, the original user permissions are restored.
- Roles can be used to provision access to almost all the AWS resources
- Permissions provided to the User through the Role can be further restricted per user by passing optional policy to the STS request. This policy cannot be used to elevate privileges beyond what the assumed role is allowed to access

Resource based Policies

- Resource based policy allows you to attach a policy directly to the resource that you want to share, instead of using a role as a proxy.
- Resource-based policy specifies who, as a Principal in the form of a list of AWS account ID numbers, can access that resource and what they can access
- With Cross-account access with a resource-based policy, User still works in the trusted account and does not have to give up her user permissions in place of the role permissions.
- User can work on the resources from both the accounts at the same time and this can be useful for scenarios for e.g. copying of objects from one bucket to the other
- Resource that you want to share are limited to resources which support resource-based policies
    - **Amazon S3 allows you to define Bucket policy to grant access to the bucket and the objects**
    - **Amazon Simple Notification Service (SNS)**
    - **Amazon Simple Queue Service (SQS)**
    - **Amazon Glacier Vaults**
    - **AWS OpsWorks stacks**
    - **AWS Lambda functions**
- Resource based policies need the trusted account to create users with permissions to be able to access the resources from the trusting account
- Only permissions equivalent to, or less than, the permissions granted to your account by the resource owning account can be delegated

AWS IAM Best Practices

To help secure AWS resources, AWS recommends the following AWSIdentity and Access Management (IAM) service – IAM Best Practices

Root Account -Don't use & Lock away access keys

- Do not use AWS Root account which has full access to all the AWS resources and services including the Billing information.
- Permissions associated with your AWS Root account cannot be restricted.
- **Do not generate the access keys, if not required**
- If already generated and not needed, delete the access keys.
- If access keys needed, rotate (change) the access key regularly
- Never share your Root account credentials or access keys, instead create IAM users or Roles to grant granular access
- **Enable AWS multifactor authentication (MFA) on your AWS account**

User – Create individual IAM users

- Don't use your AWS root account credentials to access AWS, and don't share your credentials with anyone else.
- Start by creating a IAM User with Administrator role, which has access to all resources as the Root user except to the account's security credentials
- Create individual users for anyone who needs access to your AWS account and give each user unique credentials and grant different permissions

Groups – Use groups to assign permissions to IAM users

- Instead of defining permissions for individual IAM users, create groups and define the relevant permissions for each group as per the job function, and then associate IAM users to those groups.
- Users in an IAM group inherit the permissions assigned to the group and a User can belong to multiple groups
- It is much easier to add new users, remove users and modify the permissions of a group of users.

Permission – Grant least privilege

- **IAM user, by default, is created with no permissions**
- Users should be granted LEAST PRIVILEGE as required to perform a task.
- Starting with minimal permissions and add to the permissions as required to perform the job function is far better then granting access all and trying to then tightening it down

Passwords – Enforce strong password policy for users

- Enforce user to create strong passwords and enforce them to rotate their passwords periodically
- Enable a strong password policy to define passwords requirements forcing users to create passwords with requirements like at least one capital letter, one number, how frequently it should be rotated.

• MFA – Enable MFA for privileged users

- For extra security, Enable MultiFactor Authentication (MFA) for privileged IAM users, who are allowed access to sensitive resources or APIs.

Role – Use roles for applications that run on EC2 instances

- Use roles for applications running on EC2 instances instead of creating IAM user and hard-coding the credentials within that application.
- Roles do not have a permanent set of credentials associated with it but dynamically provide temporary credentials that are automatically rotated
- Hard-coding of credentials can compromise the access and are also hard to rotate. Also, they may pose a problem in the creation of new EC2 instances through Auto-scaling and handling credential rotation.

Sharing – Delegate using roles

- Allow users from same AWS account, another AWS account, or externally authenticated users (either through any corporate authentication service or through Google, Facebook etc) to use IAM roles to specify the permissions which can then be assumed by them
- A role can be defined that specifies what permissions the IAM users in the other account are allowed, and from which AWS accounts the IAM users are allowed to assume the role

Rotation – Rotate credentials regularly

- Change your own passwords and access keys regularly and enforce it through a strong password policy. So even if a password or access key is compromised without your knowledge, you limit how long the credentials can be used to access your resources
- Access keys allows creation of 2 active keys at the same time for an user. These can be used to rotate the keys.

Track – Remove unnecessary credentials

- Remove IAM user and credentials (that is, passwords and access keys) that are not needed
- Use the Credential report that lists all IAM users in the account and status of their various credentials, including passwords, access keys, and MFA devices and usage pattern to figure out what can be removed
- Passwords and access keys that have not been used recently might be good candidates for removal.

Conditions – Use policy conditions for extra security

- Define conditions under which IAM policies allow access to a resource.
- Conditions would help provide finer access control to the AWS services and resources for e.g. access limited to specific ip range or allowing only encrypted request for uploads to S3 buckets etc.

Auditing – Monitor activity in the AWS account

- Enable logging features provided through CloudTrail, S3, CloudFront in AWS to determine the actions users have taken in the account and the resources that were used.
- Log files show the time and date of actions, the source IP for an action, which actions failed due to inadequate permissions, and more.

- ## IAM – Key Notes

  - There are 3 IAM identities: Users, Groups, and Roles.
    - **User:** A primary use for IAM users is to give people the ability to sign in to the AWS Management Console for interactive tasks and to make programmatic requests to AWS services using the API or CLI.
    - **Group:** is a collection of IAM users. You can use groups to specify permissions for a collection of users, which can make these permissions easier to manage for the users.
    - **Role:** is very similar to a user in that it is an identity with permission policies that determine what the identity can and cannot do in AWS. However, a role does not have any credentials (password or access keys) associated with it.
  - By default, a new IAM user has no password and no access key-You must create the type of credentials for an IAM user based on what the user will be doing.
    - Users need their own access keys to make programmatic calls to AWS from the AWS Command Line Interface (AWS CLI), Tools for Windows PowerShell, the AWS SDKs, or direct HTTP calls using the APIs for individual AWS services.
  - A **policy** is an entity in AWS that, when attached to an identity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Policies are stored in AWS as JSON
    - The following policy types, listed in order of frequency, are available for use in AWS:
    - **Identity-based policies** – Attach managed and inline policies to IAM identities, such as users, groups to which users belong, and roles.
    - **Resource-based policies** – Attach inline policies to resources. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies.
    - **Organizations SCPs** – Use an AWS Organizations service control policy (SCP) to apply a permissions boundary to an AWS Organizations organization or organizational unit (OU).
    - **Access control lists (ACLs)** – Use ACLs to control what principals can access a resource. ACLs are similar to resource-

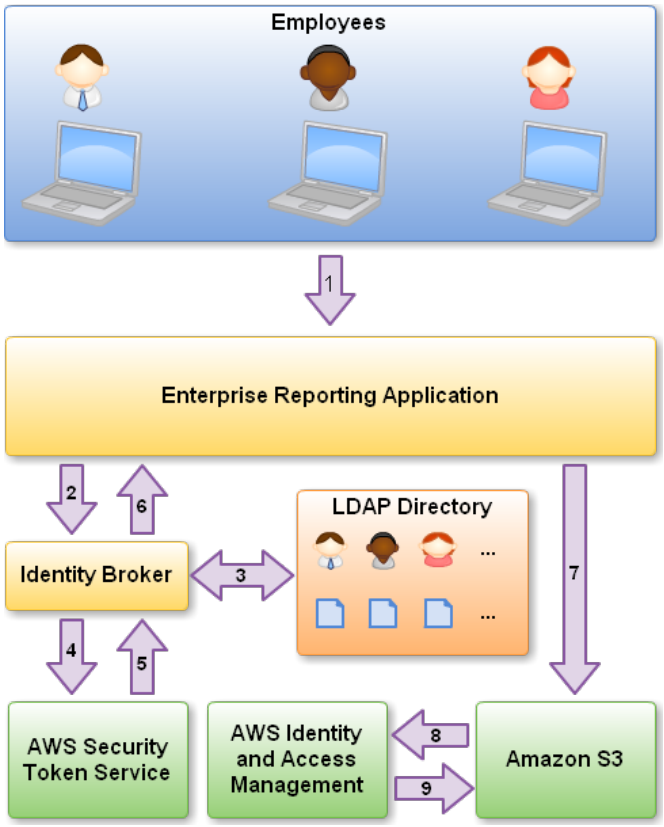
based policies, although they are the only policy type that does not use the JSON policy document structure.

- **Role:**
  - Use an IAM role to manage *temporary* credentials for applications that run on an EC2 instance. When you use an IAM role, you don't have to distribute long-term credentials (such as a user name and password or access keys) to an EC2 instance.
  - Instead, the role supplies temporary permissions that applications can use when they make calls to other AWS resources. When you launch an EC2 instance, you specify an IAM role to associate with the instance. Applications that run on the instance can then use the role-supplied temporary credentials to sign API requests.
  - With web identity federation, you don't need to create custom sign-in code or manage your own user identities. Instead, users of your app can sign in using a well-known identity provider (IdP) —such as Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP, receive an authentication token, and then exchange that token for temporary security credentials in AWS that map to an IAM role with permissions to use the resources in your AWS account. Using an IdP helps you keep your AWS account secure because you don't have to embed and distribute long-term security credentials with your application.
  - For example, create a new IAM role with permissions to access the DynamoDB table and assign it to the EC2 instances.

- **STS**: **To use temporary security credentials in code** - You can use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources.I**f you want to run AWS CLI commands or code inside an EC2 instance, the recommended way to get credentials is to use roles for Amazon EC2.**
- **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).**
  - **If you're making direct HTTPS API requests to AWS, you can sign those requests with the temporary security credentials that you get from the AWS Security Token Service (AWS STS).**
- Temporary credentials are useful in scenarios that involve identity federation, delegation, cross-account access, and IAM roles.

- **Identity Federation (SAML)**: You can enable SAML authentication for your AWS accounts by using AWS Identity and Access Management (IAM). You can add SAML support to your web and mobile apps running on the AWS Cloud by using Amazon Cognito. You can manage your user identities in an external system outside of AWS and grant users who sign in from those systems access to perform AWS tasks and access your AWS resources. IAM supports two types of identity federation. In both cases, the identities are stored outside of AWS. The distinction is where the external system resides—in your data center or an external third party on the web:
- **Enterprise identity federation** – You can authenticate users in your organization's network, and then provide those users access to AWS without creating new AWS identities for them and requiring them to sign in with a separate user name and password. This is known as the *single sign-on* (SSO) approach to temporary access. AWS STS supports open standards like Security Assertion Markup Language (SAML) 2.0, with which you can use Microsoft AD FS to leverage your Microsoft Active Directory. You can also use SAML 2.0 to manage your own solution for federating user identities:
  - **Custom federation/identy broker** – You can use your organization's authentication system to grant access to AWS resources. **(To integrate AWS Identity and Access Management with an on premise LDAP)**
    - you can develop an on-premise custom identity broker application and use STS to issue short-lived AWS credentials.
    - The application verifies that employees are signed into the existing LDAP server of the company. The identity broker application then obtains temporary security credentials for the employees through the use of the Simple Token Service. This will allow the employees access to AWS.

- Refer to the diagram and links below for reference.



  - **Federation using SAML 2.0** – You can use your organization's authentication system and SAML to grant access to AWS resources. For more information and an example scenario, see About SAML 2.0-based Federation.
  - **Web identity federation** – You can let users sign in using a well-known third party identity provider such as Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC) 2.0 compatible provider. You can exchange the credentials from that provider for temporary permissions to use resources in your AWS account. T**his is known as the *web identity federation* approach to temporary access.** When you use web identity federation for your mobile or web application, you don't need to create custom sign-in code or manage your own user identities. Using web identity federation helps you keep your AWS account secure, because you don't have to distribute long-term security credentials.
    - **AWS STS web identity federation supports** Login with Amazon, Facebook, Google, and any OpenID Connect (OIDC)-compatible identity provider.
    - **Note: For mobile applications, we recommend that you use Amazon Cognito.** Amazon Cognito supports the same identity providers as AWS STS, and also supports unauthenticated (guest) access and lets you migrate user data

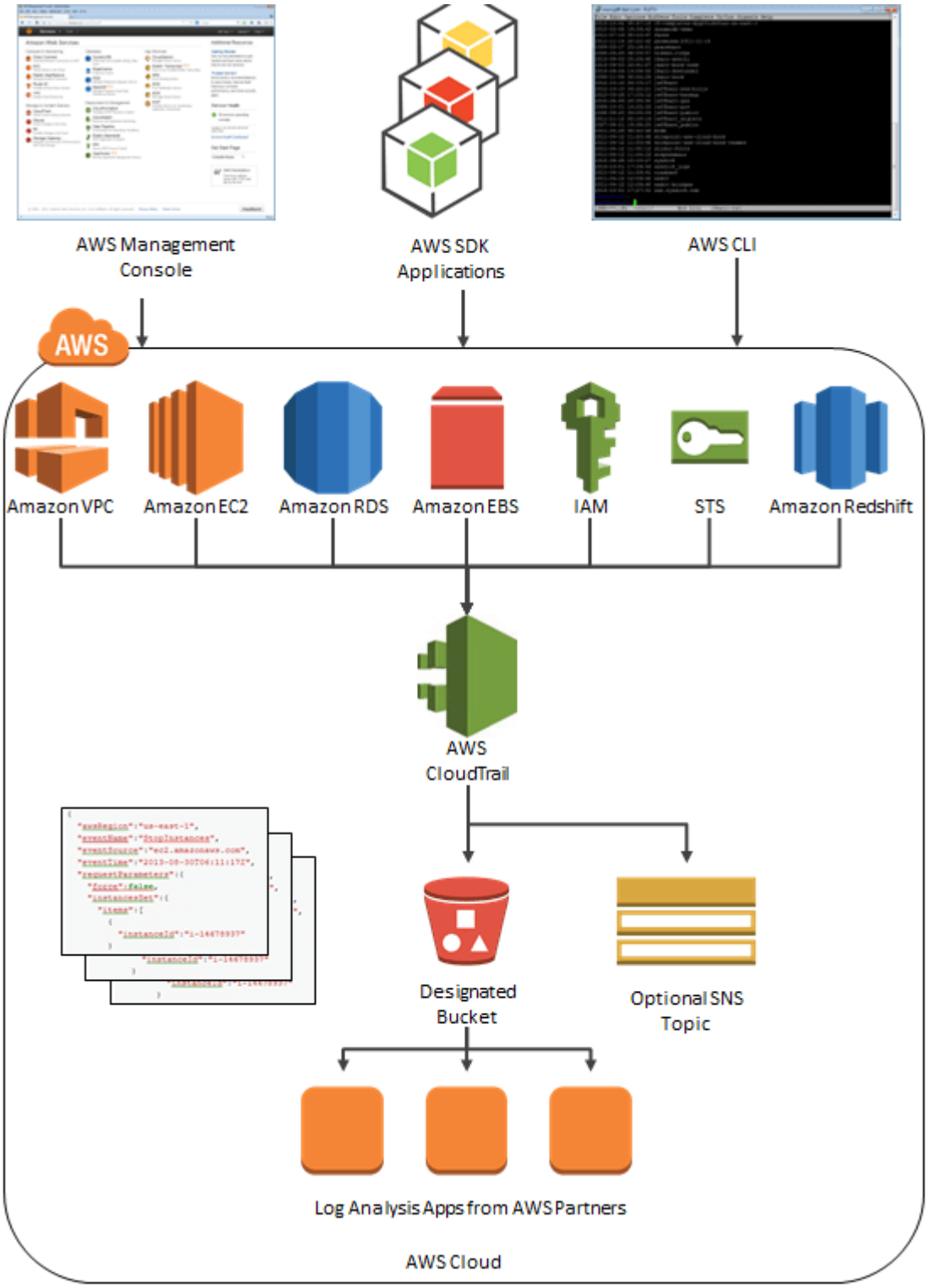
when a user signs in. Amazon Cognito also provides API operations for synchronizing user data so that it is preserved as users move between devices.

- **cross-account access**: You can use an IAM role to delegate access to resources that are in different AWS accounts that you own. You share resources in one account with users in a different account. By setting up cross-account access in this way, you don't need to create individual IAM users in each account. In addition, users don't have to sign out of one account and sign into another in order to access resources that are in different AWS accounts..
  - You can use the consolidated billing feature in AWS Organizations to consolidate payment for multiple AWS accounts or multiple AISPL accounts. With consolidated billing, you can see a combined view of AWS charges incurred by all of your accounts. You also can get a cost report for each member account that is associated with your master account. Consolidated billing is offered at no additional charge. AWS and AISPL accounts can't be consolidated together

AWS CloudTrail

- **AWS CloudTrail helps to get a history of AWS API calls and related events for the AWS account.**
- CloudTrail tracking includes calls made by using the AWS Management Console, AWS SDKs, command line tools, and higher-level AWS services (such as AWS CloudFormation)
- CloudTrail helps to identify which users and accounts called AWS for services that support CloudTrail, the source IP address the calls were made from, and when the calls occurred.
- CloudTrail is per AWS account and enabled per region for all the services supporting it
- AWS API call history produced by CloudTrail enables security analysis, resource change tracking, and compliance auditing

CloudTrail Works



- AWS CloudTrail captures AWS API calls and related events made by or on behalf of an AWS account and delivers log files to an specified S3 bucket.
- Log files contain API calls from all of the

- **Log files from all the regions can be delivered to a single S3 bucket and are encrypted, by default, using S3 server-side encryption (SSE)**

- CloudTrail typically delivers log files within 15 minutes of an API call and publishes new log files multiple times an hour, usually about every 5 mins.

- CloudTrail can be configured, optionally, to deliver events to a log group to be monitored by CloudWatch Logs.

- Amazon SNS notifications can be configured to be sent each time a log file is delivered to your bucket.

- A trail, which is a configuration, needs to be created that enables logging of the AWS API activity and related events in your account.

- Trail can be created with CloudTrail console, AWS CLI, or CloudTrail API.

- A trail can be applied to all regions or a single region:
    - **A trail that applies to all regions**
        - When a trail is created that applies to all regions, CloudTrail creates the same trail in each region, records the log files in each region, and delivers the log files to the specified single S3 bucket (and optionally to the CloudWatch Logs log group)
        - Default setting when a trail is created using the CloudTrail console
        - A single SNS topic for notifications and CloudWatch Logs log group for events would suffice for all regions
        - **Advantages**
            - configuration settings for the trail apply consistently across all regions
            - **manage trail configuration for all regions from one location.**
            - immediately receive events from a new region
            - receive log files from all regions in a single S3 bucket and optionally in a CloudWatch Logs log group.
            - create trails in regions not used often to monitor for unusual activity

    - **A trail that applies to one region**
        - A S3 bucket can be specified that receives events only from that region and it can be in any region that you specify.
        - Additional individual trails created that apply to specific regions, those trails can deliver event logs to a single S3 bucket.

- Turning on a trail' means that create a trail and start logging

- CloudTrail supports five trails per region. A trail that applies to all regions counts as one trail in every region

- As a best practice, a trail can be created that applies to all regions in the AWS partition for e.g. aws for all standard aws regions or aws-cn for china
- IAM can be used to control which AWS users can create, configure, or delete trails, start and stop logging, and access the buckets that contain log information.

- Log file integrity validation can be enabled to verify that log files have remained unchanged since CloudTrail delivered them.

Global Services Option

- For most services, events are sent to the region where the action happened.

- For global services such as IAM, AWS STS, and Amazon CloudFront, events are delivered to any trail that has the Include global services option enabled.

- AWS OpsWorks and Amazon Route 53 actions are logged in the US East (N. Virginia) region.

- To avoid receiving duplicate global service events, remember
    - Global service events are always delivered to trails that have the Apply trail to all regions option enabled.
    - Events are delivered from a single region to the bucket for the trail. This setting cannot be changed.
    - If you have a single region trail, you should enable the Include global services option.
    - If you have multiple single region trails, you should enable the Include global services option in only one of the trails.

- **About global service events**
    - **have a trail with the Apply trail to all regions option enabled.**
    - **have multiple single region trails.**
    - **do not need to enable the Include global services option for the single region trails. Global service events are delivered for the first trail.**

Validating CloudTrail Log File Integrity

- **Validated log files are invaluable in security and forensic investigations**

- **CloudTrail log file integrity validation can be used to check whether a log file was modified, deleted, or unchanged after CloudTrail delivered it**

- Validation feature is built using industry standard algorithms: SHA-256 for hashing and SHA-256 with RSA for digital signing which makes it computationally infeasible to modify, delete or forge CloudTrail log files without detection

- When log file integrity validation is enabled:-
  - CloudTrail creates a hash for every log file that it delivers.
  - Every hour, CloudTrail also creates and delivers a digest file that references the log files for the last hour and contains a hash of each.
  - CloudTrail signs each digest file using the private key of a public and private key pair.
  - After delivery, the public key can be used to validate the digest file.
  - CloudTrail uses different key pairs for each AWS region.
  - Digest files are delivered to the same Amazon S3 bucket, but a separate folder, associated with the trail for the log files
  - Separation of digest files and log files enables enforcement of granular security policies and permits existing log processing solutions to continue to operate without modification.
  - Each digest file also contains the digital signature of the previous digest file if one exists.
  - Signature for the current digest file is in the metadata properties of the digest file Amazon S3 object.
  - **Log files and digest files can be stored in Amazon S3 or Amazon Glacier securely, durably and inexpensively for an indefinite period of time.**
  - **To enhance the security of the digest files stored in Amazon S3, Amazon S3 MFA Delete can be enabled**

- **CloudTrail Enabled Use Cases**:
  - **Track changes to AWS resources**
    - Can be used to track creation, modification or deletion of AWS resources
  - **Compliance Aid**
    - easier to demonstrate compliance with internal policy and regulatory standards
  - **Troubleshooting Operational Issues**
    - identify the recent changes or actions to troubleshoot any issues
  - **Security Analysis**

        - use log files as inputs to log analysis tools to perform security analysis and to detect user behaviour patterns

CloudTrail Processing Library (CPL)

- CloudTrail Processing Library (CPL) helps build applications to take immediate action on events in CloudTrail log files

- CPL helps to
    - read messages delivered to SNS or SQS
    - downloads and reads the log files from S3 continuously
    - serializes the events into a POJO
    - allows custom logic implementation for processing
    - fault tolerant and supports multi-threading

CloudTrail – Key Notes

- Simplifies compliance audit.
- Can log API calls.
- Delivers log files to an S3 bucket.

- Can create **two types of trail:**
    - **Trail that applies to all region.**
    - **Trail that applies to one region.**

- You can get a history of AWS API calls for your account, including API calls made AWS

- **Enable Tracking and Alerting**
    - AWS CloudTrail and AWS Config are two very useful and important services. AWS CloudTrail provides an audit trail of your events with complete information, whereas AWS Config provides configuration history about your AWS Account.
- **AWS CloudTrail with CloudWatch**
    - By leveraging CloudWatch, you can set up alerts for CloudTrail events. For example, if there is a Security Group and the ingress or egress rules change, it will automatically send notifications to SNS subscriptions, alerting the security team. The security team can immediately identify the IAM user performing this change with his/her IP address and any other details.
    - You can create a metric filter by defining a filter pattern in CloudWatch Logs to identify any changes to your existing security groups:
        -
        - { ($.eventName = AuthorizeSecurityGroupIngress) || ($.eventName = AuthorizeSecurityGroupEgress) |
        - | ($.eventName = RevokeSecurityGroupIngress) || ($.eventName = RevokeSecurityGroupEgress) |

        - | ($.eventName = CreateSecurityGroup) || ($.eventName = DeleteSecurityGroup) }
    - After creating the filter, you can create a CloudWatch Alarm to notify you of any changes in your existing security groups. Additionally, you can keep these CloudWatch events in CloudTrail and set notifications associated with any of your deployment's security groups configurations.
    - **Note:** In order to avoid conflicts, you should document which users have permission to modify security groups and restrict their ability to modify CloudWatch or CloudTrail notification settings.

AWS Key Management Service – KMS (think in ansible vault working)

- AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data(create, store, enable, disable and delete)

- **Keys generated via KMS can never be exported.**

- **AWS KMS is a managed encryption service that enables encryption of data easily**

- KMS provides a highly available key storage, management, and auditing solution to encrypt the data across AWS services & within applications
- KMS is integrated with several other AWS services to make encrypting data in those service easy

- **KMS Keys are only stored and used in the region in which they are created.** They cannot be transferred to another region

- KMS enforces usage and management policies, to control which IAM user, role from your account or other accounts who can manage and use keys

- **KMS is integrated with CloudTrail, so all requests to use the keys are logged to understand who used which key when**

- **KMS allows rotation of the keys,**
    - if keys generated by KMS rotated automatically by KMS, data does not need to be re-encrypted. KMS keeps previous versions of keys to use for decryption of data encrypted under an old version of a key. All new encryption requests against a key in AWS KMS are encrypted under the newest version of the key.
    - if manually rotated, data has to be re-encrypted depending on the application's configuration
    - Automatic key rotation is not supported for imported keys

KMS Working

- KMS centrally manages and securely stores the keys
- Keys can be generated or imported from your key management infrastructure
- Keys can be used from within the applications and supported AWS services to protect the data, but the key never leaves KMS AWS.
- Data is submitted to AWS KMS to be encrypted, or decrypted, under keys that you control.
- Usage policies on these keys can be set that determine which users can use them to encrypt and decrypt data.

Envelope encryption

- AWS cloud services integrated with **AWS KMS use a method called envelope encryption to protect the data.**
- Envelope encryption is an optimized method for encrypting data that uses two different keys
- **With Envelop encryption:**
  - A data key is generated and used by the AWS service to encrypt each piece of data or resource.
  - Data key is encrypted under a master key that you define in AWS KMS.
  - Encrypted data key is then stored by the AWS service.
  - For data decryption by the AWS service, the encrypted data key is passed to AWS KMS and decrypted under the master key that was originally encrypted under so the service can then decrypt your data.
- KMS does support sending data less than 4 KB to be encrypted, envelope encryption can offer significant performance benefits
- When the data is encrypted directly with KMS it must be transferred over the network.
- Envelope encryption reduces the network load for the application or AWS cloud service as Only the request and fulfilment of the data key through KMS must go over the network

Amazon EBS Encryption

- When you attach an encrypted Amazon EBS volume to a , data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that host Amazon EC2 instances.

- This feature is supported on all . (General Purpose SSD [gp2], Provisioned IOPS SSD [io1], Throughput Optimized HDD [st1], Cold HDD [sc1], and Magnetic [standard]). You can expect the same IOPS performance on encrypted volumes as on unencrypted volumes, with a minimal effect on latency.

- You access encrypted volumes the same way you access other volumes; encryption and decryption are handled transparently and they require no additional action from you, your EC2 instance, or your application. Snapshots of encrypted volumes are automatically encrypted, and volumes that are created from encrypted snapshots are also automatically encrypted.

- The encryption status of an EBS volume is determined when you create the volume. You cannot change the encryption status of an existing volume. However, you can between encrypted and unencrypted volumes and apply a new encryption status while copying a snapshot.
  - To , select the appropriate box in the Amazon EBS section of the Amazon EC2 console. You can use a custom customer master key (CMK) by choosing one from the list that appears below the encryption box. If you do not specify a custom CMK, Amazon EBS uses the AWS-managed CMK for Amazon EBS in your account. If there is no AWS-managed CMK for Amazon EBS in your account, Amazon EBS creates one.
    - The following explains how Amazon EBS uses your CMK:
      1. When you create an encrypted EBS volume, Amazon EBS sends a request to AWS KMS, specifying the CMK that you chose for EBS volume encryption.
      2. AWS KMS generates a new data key, encrypts it under the specified CMK, and then sends the encrypted data key to Amazon EBS to store with the volume metadata.
      3. When you attach the encrypted volume to an EC2 instance, Amazon EC2 sends the encrypted data key to AWS KMS with a request.
      4. AWS KMS decrypts the encrypted data key and then sends the decrypted (plaintext) data key to Amazon EC2.
      5. Amazon EC2 uses the plaintext data key in hypervisor memory to encrypt disk I/O to the EBS volume. The data key persists in memory as long as the EBS volume is attached to the EC2 instance.

- Amazon EBS encryption offers a simple encryption solution for your EBS volumes without the need to build, maintain, and secure your own key

management infrastructure. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:
    - **Data at rest inside the volume**
    - **All data moving between the volume and the instance**
    - **All snapshots created from the volume**
    - **All volumes created from those snapshots**

- Encryption operations occur on the servers that host EC2 instances, ensuring the security of both data-at-rest and data-in-transit between an instance and its attached EBS storage.

- **Public snapshots of encrypted volumes are not supported, but you can share an encrypted snapshot with specific accounts.**

- **Amazon EBS encryption is only available on certain instance types.** You can attach both encrypted and unencrypted volumes to a supported instance type.

- Amazon EBS encryption uses AWS Key Management Service (AWS KMS) customer master keys (CMKs) when creating encrypted volumes and any snapshots created from them. A unique AWS-managed CMK is created for you automatically in each region where you store AWS assets. This key is used for Amazon EBS encryption unless you specify a customer-managed CMK that you created separately using AWS KMS.
    - Creating your own CMK gives you more flexibility, including the ability to create, rotate, and disable keys to define access controls.

- You cannot change the CMK that is associated with an existing snapshot or encrypted volume. However, you can associate a different CMK during a snapshot copy operation so that the resulting copied snapshot uses the new CMK.

- EBS encrypts your volume with a data key using the industry-standard AES-256 algorithm. Your data key is stored on-disk with your encrypted data, but not before EBS encrypts it with your CMK—it never appears there in plain-text. The same data key is shared by snapshots of the volume and any subsequent volumes created from those snapshots.

- **There is no direct way to encrypt an existing unencrypted volume**, **or to remove encryption from an encrypted volume**. However, you can migrate data between encrypted and unencrypted volumes. You can also apply a new encryption status while copying a snapshot:
    - While copying an unencrypted snapshot of an unencrypted volume, you can encrypt the copy. Volumes restored from this encrypted copy are also encrypted.
    - While copying an encrypted snapshot of an encrypted volume, you can associate the copy with a different CMK. Volumes restored from the encrypted copy are only accessible using the newly applied CMK.

- When you have access to both an encrypted and unencrypted volume, you can freely transfer data between them. EC2 carries out the encryption and decryption operations transparently.

- **How To migrate data between encrypted and unencrypted volumes**
    - Create your EBS destination volume (encrypted or unencrypted, depending on your need)
    - Attach the destination volume to the instance that hosts the data to migrate.
    - **Make the destination volume available:**
        - **EBS Volume Available for Use on Linux.**
            - For Linux instances, you can create a mount point at /mnt/destination and mount the destination volume there.
            - Copy the data from your source directory to the destination volume. It may be most convenient to use a bulk-copy utility for this.
            - Use the rsync command as follows to copy the data from your source to the destination volume. In this example, the source data is located in /mnt/source and the destination volume is mounted at /mnt/destination.
                - [ec2-user ~]$ sudo rsync -avh --progress /mnt/source/ /mnt/destination/
        - **Windows**
            - At a command prompt, use the robocopy command to copy the data from your source to the destination volume. In this example, the source data is located in D:\ and the destination volume is mounted at
                - PS C:\> robocopy D:\<sourcefolder> E:\ <destinationfolder> /e /copyall /eta
    - **Note:** IT is recommend explicitly naming folders rather than copying the entire volume in order to avoid potential problems with hidden folders.

- **Apply Encryption While Copying a Snapshot**
    - Create a snapshot of your unencrypted EBS volume. This snapshot is also unencrypted.
    - Copy the snapshot while applying encryption parameters. The resulting target snapshot is encrypted.
        - Restore the encrypted snapshot to a new volume, which is also encrypted.

- **Encrypt a Snapshot to a New CMK**
    - The ability to encrypt a snapshot during copying also allows you to apply a new CMK to an already-encrypted snapshot that you own. Volumes restored from the resulting copy are only accessible using the new CMK.
    - **Note:** If you copy a snapshot to a new CMK, a complete (non-incremental) copy will always be created, resulting in additional storage costs.

- In a related scenario, you may choose to apply new encryption parameters to a copy of a snapshot that has been shared with you. Before you can restore a volume from a shared encrypted snapshot, you must create your own copy of it. By default, the copy is encrypted with a CMK shared by the snapshot's owner. However, we recommend that you create a copy of the shared snapshot using a different CMK that you control. This protects your access to the volume if the original CMK is compromised, or if the owner revokes the CMK for any reason.

Amazon S3 - Server-Side Encryption (at rest): Using SSE-KMS

This topic discusses how to protect data at rest within Amazon S3 by using AWS KMS. **There are two ways to use AWS KMS with Amazon S3**. You can **use server-side encryption** to protect your data with a **customer master key** or you can use a **AWS KMS customer master key** with the Amazon S3 encryption client to protect your data on the client side.

You can protect data at rest in Amazon S3 by using **three different modes of server-side encryption: SSE-S3, SSE-C, or SSE-KMS.**

- **SSE-S3 requires that Amazon S3 manage the data and master encryption keys. (SSE-S3)**
- **SSE-C requires that you manage the encryption key. SSE-C (Customer master key)**
- **SSE-KMS requires that AWS manage the data key but you manage the master key in AWS KMS. (SSE-KMS).**

You can request encryption and the master key you want by using the Amazon S3 console or API. In the console, check the appropriate box to perform encryption and select your key from the list. For the Amazon S3 API, specify encryption and choose your key by setting the appropriate headers in a GET or PUT request.

You can choose a specific customer-managed master key or accept the AWS-managed key for Amazon S3 under your account. If you choose to encrypt your data, AWS KMS and Amazon S3 perform the following actions:

- Amazon S3 requests a plain-text data key and a copy of the key encrypted by using the specified customer-managed master key or the AWS-managed master key.
- AWS KMS creates a data key, encrypts it by using the master key, and sends both the plain-text data key and the encrypted data key to Amazon S3.
- Amazon S3 encrypts the data using the data key and removes the plaintext key from memory as soon as possible after use.
- Amazon S3 stores the encrypted data key as metadata with the encrypted data.

Amazon S3 and AWS KMS perform the following actions when you request that your data be decrypted:

- Amazon S3 sends the encrypted data key to AWS KMS.
- AWS KMS decrypts the key by using the appropriate master key and sends the plaintext key back to Amazon S3.
- Amazon S3 decrypts the ciphertext and removes the plaintext data key from memory as soon as possible.

Using the Amazon S3 Encryption Client

- **You can use the Amazon S3 encryption client in the AWS SDK from your own application to encrypt objects and upload them to Amazon S3. This method allows you to encrypt your data locally to ensure its security as it passes to the Amazon S3 service.** The S3 service receives your encrypted data and does not play a role in encrypting or decrypting it.

- The Amazon S3 encryption client encrypts the object by using envelope encryption. The client calls AWS KMS as a part of the encryption call you make when you pass your data to the client. AWS KMS verifies that you are authorized to use the customer master key and, if so, returns a new plaintext data key and the data key encrypted under the customer master key. The encryption client encrypts the data by using the plaintext key and then deletes the key from memory. The encrypted data key is sent to Amazon S3 to store alongside your encrypted data.

- **Encryption Context**
  - Each service that is integrated with AWS KMS specifies an encryption context when requesting data keys, encrypting, and decrypting. The encryption context is additional authenticated information that AWS KMS uses to check for data integrity. That is, when an encryption context is specified for an encryption operation, the service also specifies it for the decryption operation or decryption will not succeed. If you are using SSE-KMS or the Amazon S3 encryption client to perform encryption, Amazon S3 uses the bucket path as the encryption context. In the requestParameters field of a CloudTrail log file, the encryption context will look similar to this.

- **Some keys**
  - One critical aspect of using any AWS REST API call is the signature used to authenticate the request. This signature is vital if you wish to allow only authenticated users to access your data.
- Using KMS you have access control so you are able to ensure that only permitted users are able to access the encryption keys needed to decrypt data. You control these access permissions in KMS by using policies.

- With KMS, you also can see when, where, and by whom your customer managed keys (CMK) are used, because all API calls are logged by AWS CloudTrail. These logs provide you with full audit capabilities for your keys. There is also one default CMK for each account and service integrated with KMS. This key is referred to as the default service key and will always be

listed with an alias of aws/[servicename] (for example, aws/s3) in your KMS console. This default service key is used if you choose not to create your own custom keys in KMS.

Amazon Redshift Encryption

- An Amazon Redshift data warehouse is a collection of computing resources called nodes, which are organized into a group called a cluster. Each cluster runs an Amazon Redshift engine and contains one or more databases.

- Amazon Redshift uses a four-tier, key-based architecture for encryption. The architecture consists of data encryption keys, a database key, a cluster key, and a master key.

- **Data encryption keys encrypt data blocks in the cluster.** Each data block is assigned a randomly-generated AES-256 key. **These keys are encrypted by using the database key for the cluster.**

- The database key encrypts data encryption keys in the cluster. **The database key is a randomly-generated AES-256 key. It is stored on disk in a separate network from the Amazon Redshift cluster and passed to the cluster across a secure channel.**

- **The cluster key encrypts the database key for the Amazon Redshift cluster. You can use AWS KMS, AWS CloudHSM, or an external hardware security module (HSM) to manage the cluster key.**

- If the master key resides in AWS KMS, it encrypts the cluster key. You can request encryption by checking the appropriate box in the Amazon Redshift console. You can specify a customer-managed master key to use by choosing one from the list that appears below the encryption box. If you do not specify a customer-managed key, the AWS-managed key for Amazon Redshift under your account will be used.

Amazon Redshift Database Encryption

- **In Amazon Redshift, you can enable database encryption for your clusters to help protect data at rest. When you enable encryption for a cluster, the data blocks and system metadata are encrypted for the cluster and its snapshots.**

- Encryption is an optional, immutable setting of a cluster. If you want encryption, you enable it during the cluster launch process. To go from an unencrypted cluster to an encrypted cluster or the other way around, unload your data from the existing cluster and reload it in a new cluster with the chosen encryption setting

- Though encryption is an optional setting in Amazon Redshift, we recommend **enabling it for clusters that contain sensitive data**. Additionally, you might be required to use encryption depending on the guidelines or regulations that govern your data. For example, the Payment Card Industry Data Security Standard (PCI DSS), the Sarbanes-Oxley Act (SOX), the Health Insurance Portability and Accountability Act (HIPAA), and other such regulations provide guidelines for handling specific types of data.

- **Amazon Redshift uses a hierarchy of encryption keys to encrypt the database. You can use either AWS Key Management Service (AWS KMS) or a hardware security module (HSM) to manage the top-level encryption keys in this hierarchy. The process that Amazon Redshift uses for encryption differs depending on how** you manage keys.

- Additionally, **Amazon Redshift automatically integrates with AWS KMS but not with an HSM**. **When you use an HSM, you must use client and server certificates to configure a trusted connection between Amazon Redshift and your HSM.**

Database Encryption for Amazon Redshift Using AWS KMS

- When you choose AWS KMS for key management with Amazon Redshift, there is **a four-tier hierarchy of encryption keys**. These keys, **in hierarchical order,** are the **master key, a cluster encryption key (CEK), a database encryption key (DEK), and data encryption keys.**

- **When you launch your cluster, Amazon Redshift returns a list of the customer master keys (CMKs)** that your AWS account has created or has permission to use in AWS KMS. You select a CMK to use as your master key in the encryption hierarchy.

- Your default key is an AWS-managed key that is created for your AWS account to use in Amazon Redshift. AWS KMS creates this key the first time you launch an encrypted cluster in a region and choose the default key.

- **If you don’t want to use the default key, you must have (or create) a customer-managed CMK separately in AWS KMS before you launch your cluster in Amazon Redshift.** Customer-managed CMKs give you more flexibility, including the ability to create, rotate, disable, define access control for, and audit the encryption keys used to help protect your data.

- **If you want to use a AWS KMS key from another AWS account, you must have permission to use the key and specify its ARN in Amazon Redshift.**

- **After you choose a master key, Amazon Redshift requests that AWS KMS generate a data key and encrypt it using the selected master key. This data key is used as the CEK in Amazon Redshift**. **AWS KMS exports the encrypted CEK to Amazon Redshift, where it is stored internally on disk in a separate network from the cluster** along with the grant to the CMK and the encryption context for the CEK. **Only the encrypted CEK is exported to Amazon Redshift; the CMK remains in AWS KMS**. Amazon Redshift also passes the encrypted **CEK over a secure channel to the cluster and loads it into memory**. **Then, Amazon Redshift calls AWS KMS to decrypt the CEK and loads the decrypted CEK into memory.**

- Next, Amazon Redshift randomly generates a key to use as the DEK and loads it into memory in the cluster. The decrypted CEK is used to encrypt the DEK, which is then passed over a secure channel from the cluster to be stored internally by Amazon Redshift on disk in a separate network from the cluster. Like the CEK, both the encrypted and decrypted versions of the DEK are loaded into memory in the cluster. The decrypted version of the DEK is then used to encrypt the individual encryption keys that are randomly generated for each data block in the database.

- **When the cluster reboots, Amazon Redshift starts with the internally stored, encrypted versions of the CEK and DEK, reloads them into memory, and then calls AWS KMS to decrypt the CEK with the CMK again** so it can be loaded into memory. The decrypted CEK is then used to decrypt the DEK again, and the decrypted DEK is loaded into memory and used to encrypt and decrypt the data block keys as needed.

KMS – Key notes

- **Server-side encryption is about protecting data at rest**. AWS Key Management Service (AWS KMS) is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud.

- AWS KMS uses customer master keys (CMKs) to encrypt your Amazon S3 objects, EBS volumes or the DataBase Key for the Redshift clusters. You use AWS KMS via the Encryption Keys section in the IAM console or via AWS KMS APIs to centrally create encryption keys, define the policies that control how keys can be used, and audit key usage to prove they are being used correctly. You can use these keys to protect your data in Amazon S3 buckets.

- The first time you add an SSE-KMS–encrypted object to a bucket in a region, a **default CMK** is created for you automatically. This key is used for SSE-KMS encryption unless you select a CMK that you created separately using AWS Key Management Service. Creating your **own CMK** gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.

- **AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data** (create, store, enable, disable and delete)

- **Keys generated via KMS can never be exported.**

- Create keys with a unique alias and description
- Import your own keys

- Control which IAM users and roles can manage keys

- Control which IAM users and roles can use keys to encrypt & decrypt data

- Choose to have AWS KMS automatically rotate keys on an annual basis

- Temporarily disable keys so they cannot be used by anyone

- Re-enable disabled keys

- Delete keys that you no longer use

- Audit use of keys by inspecting logs in AWS CloudTrail

AWS CloudHSM

- **AWS CloudHSM provides secure cryptographic key storage to customers by making hardware security modules (HSMs) available in the AWS cloud**

- AWS CloudHSM helps meet corporate, contractual and regulatory compliance requirements for data security by using dedicated HSM appliances within the AWS cloud.

- **A hardware security module (HSM)**
    - is a hardware appliance that provides secure key storage and cryptographic operations within a tamper-resistant hardware module.
    - are designed with physical and logical mechanisms, to securely store cryptographic key material and use the key material without exposing it outside the cryptographic boundary of the appliance.
    - physical protections include tamper detection and tamper response. When a tampering event is detected, the HSM is designed to securely destroy the keys rather than risk compromise
    - logical protections include role-based access controls that provide separation of duties

- CloudHSM allows encryption keys protection within HSMs, designed and validated to government standards for secure key management.

- CloudHSM helps comply with strict key management requirements within the AWS cloud without sacrificing application performance

- **CloudHSM uses SafeNet Luna SA HSM appliances**

- **HSMs are located in AWS data centers, managed and monitored by AWS, but AWS does not have access to the keys**

- **AWS can’t help recover the key material if the credentials are lost**

- **HSMs are inside your VPC and isolated from the rest of the network**

- CloudHSM provides single tenant dedicated access to each HSM appliance

- Placing HSM appliances near your EC2 instances decreases network latency, which can improve application performance

- Only you have access to the keys and operations to generate, store and manage on the keys

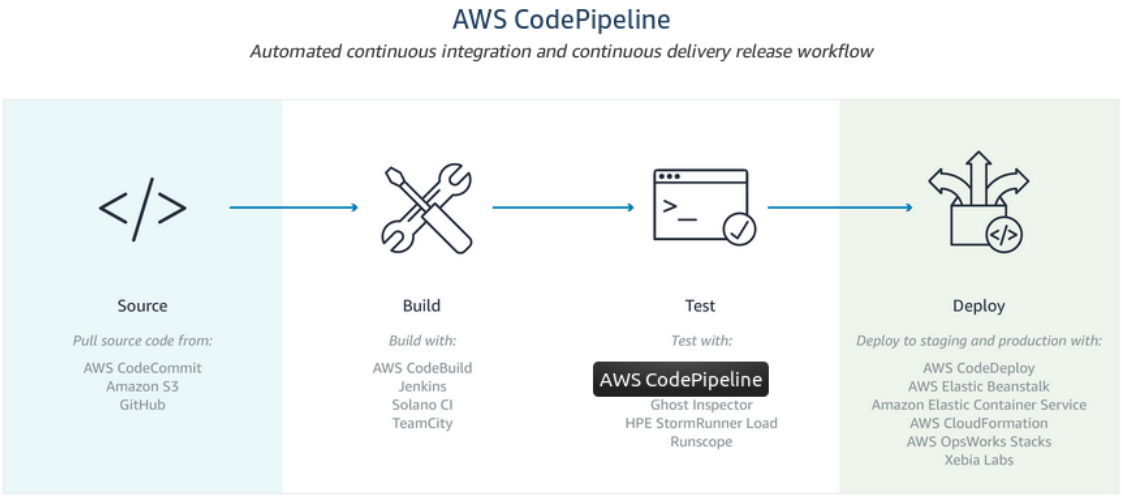
- **Integrated with Amazon Redshift and Amazon RDS for Oracle**

- **Other use cases like EBS volume encryption and S3 object encryption and key management can be handled by writing custom applications and integrating them with CloudHSM**

- Use cases: Offload SSL/TLS processing for Web Servers. Protect Private Keys, Enable Transparent Data Encrypted for databases. Government-Validated Cryptography.

AWS CodePipeline

- AWS CodePipeline is a and service for fast and reliable application and infrastructure updates.
-
- **CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models you define**. This enables you to rapidly and reliably deliver features and updates. **You can easily build out an end-to-end solution by using our pre-built plugins for popular third-party services like GitHub or integrating your own custom plugins into any stage of your release process**. With AWS CodePipeline, you only pay for what you use. There are no upfront fees or long-term commitments.



AWS Trusted Advisor

Trusted Advisor inspects the AWS environment to make recommendations for system performance, saving money, availability and closing security gaps

Trusted Advisor checks the following four categories:

- **Cost Optimization**
  - Recommendations that can potentially save money by highlighting unused resources and opportunities to reduce your bill.
- **Security**
  - Identification of security settings and gaps, in-line with the best practices, that could make the AWS solution less secure
- **Fault Tolerance**
  - Recommendations that help increase the resiliency and availability of the AWS solution by highlighting redundancy shortfalls, current service limits, and over-utilized resources.
- **Performance**
  - Recommendations that can help to improve the speed and responsiveness of the applications
- **Service Limits (– Latest Addition)**
  - Checks for service usage that is more than 80% of the service limit.
  - Values are based on a snapshot, so the current usage might differ.
  - Limit and usage data can take up to 24 hours to reflect any changes

Key: AWS Trusted Advisor analyzes your AWS environment and provides best practice recommendations in these five categories: **C**ost Optimization, **P**erformance, **F**ault Tolerance, **S**ecurity, and **S**ervice Limits.

AWS Data Pipeline

- **AWS Data Pipeline is a webservie to move data between different AWS Compute and Storage Services**
- **To automate and schedule regular data movement and data processing activities in AWS**
- AWS Data Pipeline help define data-driven workflows
- AWS Data Pipeline integrates with on-premises and cloud-based storage systems to allow developers to use their data when they need it, where they want it, and in the required format.
- AWS Data Pipeline allows you to quickly define a pipeline, which defines a dependent chain of data sources, destinations, and predefined or custom data processing activities
- Based on a defined schedule, the pipeline regularly performs processing activities such as distributed data copy, SQL transforms, EMR applications, or custom scripts against destinations such as S3, RDS, or DynamoDB.
- By executing the scheduling, retry, and failure logic for the workflows as a highly scalable and fully managed service, Data Pipeline ensures that the pipelines are robust and highly available.

AWS Data Pipeline features

- Distributed, fault-tolerant and highly available
- Managed work-flow orchestration service for data-driven workflows
- Infrastructure management service, will provision and terminate resources as required
- Provides dependency resolution
- Can be scheduled
- Grants control over retries, including frequency and number
- Native integration with S3, DynamoDB, RDS, EMR, EC2 and Redshift
- Support for both AWS based and external on-premise resources

AWS Data Pipeline Concepts

Pipeline Definition

- Pipeline definition helps the business logic to be communicated to the AWS Data Pipeline
- Pipeline definition defines the location of data (Data Nodes), activities to be performed, the schedule, resources to run the activities, per-conditions and actions to be performed

Pipeline Components, Instances, and Attempts

- Pipeline components represent the business logic of the pipeline and are represented by the different sections of a pipeline definition.
- Pipeline components specify the data sources, activities, schedule, and preconditions of the workflow
- When AWS Data Pipeline runs a pipeline, it compiles the pipeline components to create a set of actionable instances and contains all the information needed to perform a specific task
- Data Pipeline provides a durable and robust data management as it retries a failed operation depending on frequency & defined number for retries

Task Runners

- A task runner is an application that polls AWS Data Pipeline for tasks and then performs those tasks
- When Task Runner is installed and configured,
    - it polls AWS Data Pipeline for tasks associated with activated pipelines
    - after a task is assigned to Task Runner, it performs that task and reports its status back to AWS Data Pipeline.
- A task is a discreet unit of work that the Data Pipeline service shares with a task runner and differs from a pipeline, which defines activities and resources that usually yields several tasks
- Tasks can be executed either on the AWS Data Pipeline managed or user managed resources

Data Nodes

- Data Node defines the location and type of data that a pipeline activity uses as source (input) or destination (output)
- Data pipeline supports S3, Redshift, DynamoDB and SQL data nodes

Databases

- Data Pipeline supports JDBC, RDS and Redshift database

Activities

- An activity is a pipeline component that defines the work to perform
- Data Pipeline provides pre defined activities for common scenarios like sql transformation, data movement, hive queries etc
- Activities are extensible and can be used to run own custom scripts to support endless combinations

Preconditions

- Precondition is a pipeline component containing conditional statements that must be satisfied (evaluated to True) before an activity can run
- A pipeline supports
    - System-managed preconditions
        - are run by the AWS Data Pipeline web service on your behalf and do not require a computational resource
        - Includes source data and keys check for e.g. DynamoDB data, table exists or S3 key exists or prefix not empty
    - User-managed preconditions
        - run on user defined and managed computational resources
        - Can be defined as Exists check or Shell command

Resources

- A resource is the computational resource that performs the work that a pipeline activity specifies
- Data Pipeline supports AWS Data Pipeline-managed and self-managed resources
- AWS Data Pipeline-managed resources include EC2 and EMR, which are launched by the Data Pipeline service only when they’re needed
- Self managed on-premises resources can also be used, where a Task Runner package is installed which continuously polls the AWS Data Pipeline service for work to perform
- Resources can run in the same region as their working data set or even on a region different than AWS Data Pipeline
- Resources launched by AWS Data Pipeline are counted within the resource limits and should be taken into account

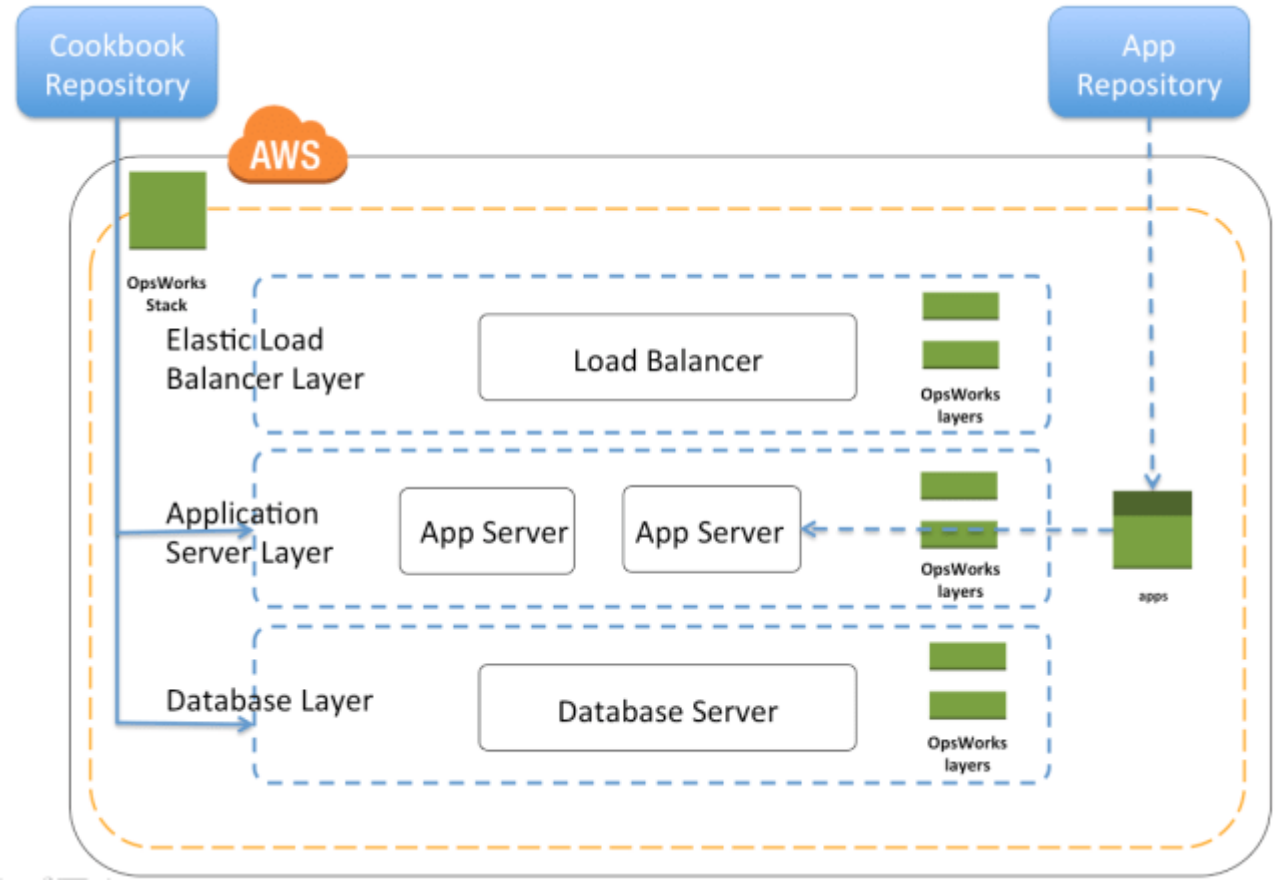
Actions

- Actions are steps that a pipeline takes when a certain event like success, failure occurs.
- Pipeline supports SNS notifications and termination action on resources

- **AWS OpsWorks** (basically, a Puppet or Chef server managed and integrated by AWS to run/deploy your code and automations)

  - AWS OpsWorks is a configuration management service that helps to configure and operate applications in a cloud enterprise by using Chef or Puppet.

OpsWorks Stacks



  - OpsWorks Stacks provides a simple and flexible way to create and manage stacks, groups of AWS resources like load balancers, web, application and database servers, and application deployed on them

  - OpsWorks Stacks helps deploy and monitor applications in the stacks.

  - Unlike OpsWorks for Chef/Puppet Automate, OpsWorks Stacks does not require or create Chef/Puppet servers; and performs some of the work of a Chef server itself

  - OpsWorks Stacks monitors instance health, and provisions new instances, when necessary, by using Auto Healing and Auto Scaling

- OpsWorks Stacks integrates with IAM to control how users can interact with stacks, what stacks can do on the users behalf, what AWS resources an app can access etc

- OpsWorks Stacks integrates with CloudWatch and CloudTrail to enable monitoring and logging

- OpsWorks Stacks can be accessed globally and can be used to create and manage instances globally

Stacks

- Stack is the core AWS OpsWorks Stacks component.
- Stack is a container for AWS resources like EC2, RDS instances etc that have a common purpose and should be logically managed together
- Stack helps manage the resources as a group and also defines some default configuration settings, such as the instances' OS and AWS region
- Stacks can also be run in VPC to be isolated from direct user interaction
- Separate Stacks can be created for different environments like Dev, QA etc

Layers

- Stacks help manage cloud resources in specialized groups called layers.
- A layer represents a set of EC2 instances that serve a particular purpose, such as serving applications or hosting a database server.
- Layers depend on Chef recipes to handle tasks such as installing packages on instances, deploying apps, and running scripts
- Custom recipes and related files is packaged in one or more cookbooks and stored in a cookbook repository such S3 or Git

Recipes and LifeCycle Events

- Layers depend on Chef recipes to handle tasks such as installing packages on instances, deploying apps, running scripts, and so on.

- OpsWorks Stacks runs the recipes for each layer, even if the instance belongs to multiple layers for e.g. instance hosting both the application and the mysql server

- AWS OpsWorks Stacks features is a set of lifecycle events – Setup, Configure, Deploy, Undeploy, and Shutdown – which automatically runs specified set of recipes at the appropriate time on each instance
    - **Setup**
        - Once a new instance has booted, OpsWorks triggers the Setup event, which runs recipes to set up the instance according to the layer configuration for e.g. installation of apache, PHP packages
        - Once setup is complete, AWS OpsWorks triggers a Deploy event, which runs recipes to deploy your application to the new instance.
    - **Configure**
        - Whenever an instance enters or leaves the online state, AWS OpsWorks triggers a Configure event on all instances in the stack.

        - Event runs each layer's configure recipes to update configuration to reflect the current set of online instances for e.g. the HAProxy layer's Configure recipes can modify the load balancer configuration to reflect any added or removed application server instances.

    - **Deploy**
        - OpsWorks triggers a Deploy event when the Deploy command is executed, to deploy the application to a set of application servers.
        - Event runs recipes on the application servers to deploy application and any related files from its repository to the layer's instances.

    - **Undeploy**
        - OpsWorks triggers an Undeploy event when an app is deleted or Undeploy command is executed to remove an app from a set of application servers.
        - Event runs recipes to remove all application versions and perform any additional cleanup tasks.

    - **Shutdown**
        - OpsWorks triggers a Shutdown event when an instance is being shut down, but before the underlying EC2 instance is actually terminated.
        - Event runs recipes to perform cleanup tasks such as shutting down services.
        - OpsWorks allows Shutdown recipes a configurable amount of time to perform their tasks, and then terminates the instance .

Instance

- An instance represents a single computing resource for e.g. EC2 instance and it defines resource's basic configuration, such as OS and size

- OpsWorks Stacks create instances and adds them to a layer.

- When the instance is started, OpsWorks Stacks launches an EC2 instance using the configuration settings specified by the instance and its layer.

- After the EC2 instance has finished booting, OpsWorks Stacks installs an agent that handles communication between the instance and the service and runs the appropriate recipes in response to lifecycle events

- OpsWorks Stacks supports instance auto-healing, whereby if an agent stops communicating with the service, OpsWorks Stacks automatically stops and restarts the instance

- OpsWorks Stacks supports the following instance types
    - 24/7 instances – launched and stopped manually
    - Time based instances – run on scheduled time

        - Load based instances – automatically started and stopped based on configurable load metrics

- Linux based computing resources created outside of the OpsWorks stacks for e.g. console or CLI can be added, incorporated and controlled through OpsWorks

Apps

- An AWS OpsWorks Stacks app represents code that you want to run on an application server residing in the app repository like S3
- App contains the information required to deploy the code to the appropriate application server instances.
- When you deploy an app, AWS OpsWorks Stacks triggers a Deploy event, which runs the Deploy recipes on the stack's instances.
- OpsWorks supports the ability to deploy multiple apps per stack and per layer

AWS Certificate Manager

- **AWS Certificate Manager** is a service that lets you easily **provision, manage, and deploy public and private** Secure Sockets Layer/Transport Layer Security **(SSL/TLS) certificates** for use with AWS services and your internal connected resources.

- **SSL/TLS certificates are used to secure network communications and establish the identity of websites over the Internet as well as resources on private networks**.

- AWS Certificate Manager removes the time-consuming manual process of purchasing, uploading, and renewing SSL/TLS certificates.

- With AWS Certificate Manager, you can quickly request a certificate, deploy it on ACM-integrated AWS resources, such as Elastic Load Balancers, Amazon CloudFront distributions, and APIs on API Gateway, and let AWS Certificate Manager handle certificate renewals. I

- t also enables you to create private certificates for your internal resources and manage the certificate lifecycle centrally. Public and private certificates provisioned through AWS Certificate Manager **for use with AWS Certificate Manager (ACM)-integrated services are free.** You pay only for the AWS resources you create to run your application. For **private certificates**, you **pay monthly for the operation of the private CA and for the private certificates you issue.**

- **Benefits and Key Features**

    - Protect and Secure Your Website
        - SSL, and its successor TLS, are industry standard protocols for encrypting network communications and establishing the identity of

websites over the Internet. SSL/TLS provides encryption for sensitive data in transit and authentication using SSL/TLS certificates to establish the identity of your site and secure connections between browsers and applications and your site. AWS Certificate Manager provides an easy way to provision and manage these certificates so you can configure a website or application to use the SSL/TLS protocol.

- Protect and Secure Your Internal Resources
  - Private certificates are used for identifying and securing communication between connected resources on private networks, such as servers, mobile and IoT devices, and applications. AWS Certificate Manager (ACM) Private Certificate Authority (CA) is a managed private CA service that helps you easily and securely manage the lifecycle of your private certificates. ACM Private CA provides you a highly-available private CA service without the upfront investment and ongoing maintenance costs of operating your own private CA. ACM Private CA extends ACM's certificate management capabilities to private certificates, enabling you to create and manage public and private certificates centrally. ACM Private CA allows developers to be more agile by providing them APIs to create and deploy private certificates pro grammatically. You also have the flexibility to create private certificates for applications that require custom certificate lifetimes or resource names. Learn more about .

- Get Certificates Easily
  - AWS Certificate Manager removes many of the time-consuming and error-prone steps to acquire an SSL/TLS certificate for your website or application. There is no need to generate a key pair or certificate signing request (CSR), submit a CSR to a Certificate Authority, or upload and install the certificate once received. With a few clicks in the AWS Management Console, you can request a trusted SSL/TLS certificate from AWS. Once the certificate is created, AWS Certificate Manager takes care of deploying certificates to help you enable SSL/TLS for your website or application.

- **Free Public Certificates for ACM-integrated Services**

    - With AWS Certificate Manager, there is no additional charge for provisioning public or private SSL/TLS certificates you use with , such as:
        - **Elastic Load Balancing**
        - **CloudFront**
        - **Elastic Beanstalk**
        - **API Gateway**
        - **CloudFormation**

    - you pay for the AWS resources you create to run your application. For private certificates, ACM Private CA provides you the ability to pay monthly for the service and certificates you create. You pay less per certificate as you create more private certificates.

- Managed Certificate Renewal
    - AWS Certificate Manager manages the renewal process for the certificates managed in ACM and used with ACM-integrated services, such as Elastic Load Balancing and API Gateway. ACM can automate renewal and deployment of these certificates. With ACM Private CA APIs, ACM enables you to automate creation and renewal of private certificates for on-premises resources, EC2 instances, and IoT devices.

- Secure Key Management
    - AWS Certificate Manager is designed to protect and manage the private keys used with SSL/TLS certificates. Strong encryption and key management best practices are used when protecting and storing private keys.

- Centrally Manage Certificates on the AWS Cloud
    - You will find it easy to centrally manage AWS Certificate Manager SSL/TLS certificates provided by AWS Certificate Manager in an AWS Region from the AWS Management Console, AWS CLI, or AWS Certificate Manager APIs. You can also audit the use of each certificate by reviewing your Amazon CloudTrail logs.

- Integrated with Other AWS Cloud Services
    - AWS Certificate Manager is integrated with other AWS services, so you can provision an SSL/TLS certificate and deploy it with your Elastic Load Balancer, Amazon CloudFront distribution or API in Amazon API Gateway. A**WS Certificate Manager also works with AWS Elastic Beanstalk and AWS CloudFormation for public email-validated certificates to help you manage public certificates and use them with your applications in the AWS Cloud**. To deploy a certificate with an AWS resource, you simply select the certificate you want from a drop-down list in the AWS Management Console. Alternatively, you can call an AWS API or CLI to associate the certificate with your resource. AWS Certificate Manager then deploys the certificate to the selected resource for you.
- Import Third-Party Certificates

- AWS Certificate Manager makes it easy to import SSL/TLS certificates issued by third-party Certificate Authorities (CAs) and deploy them with your Elastic Load Balancers, Amazon CloudFront distributions and APIs on Amazon API Gateway. You can monitor the expiration date of an imported certificate, and import a replacement when the existing certificate is nearing expiration. Alternatively, you can request a free certificate from AWS Certificate Manager and let AWS manage future renewals for you. Importing certificates doesn't cost anything.

Popular AWS Certificate Manager Use Cases

- **Help Meet Compliance Requirements (encryption data in transit)**
  - By making it easy to enable SSL/TLS, AWS Certificate Manager can help your organization meet regulatory and compliance requirements for encryption of data in transit. For specific information about compliance, refer to the .

- **Improved Uptime**
  - AWS Certificate Manager helps manage the challenges of maintaining SSL/TLS certificates, including certificate renewals so you don’t have to worry about expiring certificates.

AWS WAF (Web Application Firewall)

- AWS WAF is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources. AWS WAF gives you control over which traffic to allow or block to your web applications by defining customizable web security rules.

- You can use AWS WAF to create custom rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that are designed for your specific application. New rules can be deployed within minutes, letting you respond quickly to changing traffic patterns. Also, AWS WAF includes a full-featured API that you can use to automate the creation, deployment, and maintenance of web security rules.

Some Security knowledge:

- **Amazon Inspector**
  - Is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for vulnerabilities or deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed list of security findings prioritized by level of severity. These findings can be reviewed directly or as part of detailed assessment reports which are available via the Amazon Inspector console or API.

    - To help you get started quickly, Amazon Inspector includes a knowledge base of hundreds of rules mapped to common security best practices and vulnerability definitions. Examples of built-in rules include checking for remote root login being enabled, or vulnerable software versions installed. These rules are regularly updated by AWS security researchers.

- **Penetration tests**
    - Penetration tests are allowed after obtaining permission from aws to perform them
    -
- **Credentials reports**
    - Credential reports generated in CSV

- **Best practice:**
    - make effective policies as for example on S3, defines a policy in which data is only deleted after a user has been validated this operation through MFA correctly (for example through google authenticator)

Well architected pillars – White papper

(!) DR: means **Disaster Recovery**

The Operational Excellence pillar

Key services:

- the main key service is aws cloudFormation for Operational Excellence pillar - you can have infrastructure as code

Areas:

- In prepare :
  aws config (inventory) and config rules. It can be use to create standards for workloads and determinate if environments are compliant with those standards before putting on prod
- In operate:
  cloudWatch to monitor the operational healthy workload
- In evolve :
  Elasticsearch allows you analyze your log data

The Security pillar

Key services:

- IAM service

- Identity and Access Management (Who can do what)
  AWS IAM AWS Organizations MFA Token Temporary Security Credential Amazon Cognito to easily integrate with major login providers for authentication as Facebook Google Amazon

- Detective controls
  Aws Cloudtrail AWSConfig CloudWatch

- Infrastructure a protection
  VPC Inspector Shield WAF
  - Security Groups are built-in stateful firewalls
  - Divide layers of the stack into subnets (Public: Web Server | Private: DB)
  - Use a bastion host for access
  - Implement host based controls (Iptables)
  - ACLS
  - IAM userpolicies
  - S3 access policies

- Data protection
  Best practice of Data protection: **Encrypt the data in transit and at rest**
  Macie (protects sensitive data) AWS KMS (Protecting the keys from unauthorized access which integrates with many AWS services) **S3 (include encryption capabilities to protect the data in transit and at rest**) **EBS(include encryption capabilities to protect the data in**

transit and at rest) Elastic Load Balancing **RDS(include encryption capabilities to protect the data in transit and at rest)**

- Incident response
  AWS IAM AWS Cloudformation

The Reliability pillar (time you take to recover your services)

The ability of a system to recover from infrastructure or services failures, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.

The main Key Service is CloudWatch:

The focus areas include:

- Foundations (Manage your AWS Service limits and plan your network topology on AWS)

  AWS IAM VPC Trusted Advisor (Real time Guidance) AWS Shield(Ddos protection service)

  High Availavility:
  - No single point of failure
  - Multiple availability zones
  - Load Balancing
  - AutoScaling and healing
  - Implement software Resilence
  - Redundant connectivity

- **Change Management**
  Be aware of how changes effect a system to plan proactively
  Monitor to quickly identify trends
  - Monitor behavior SNS Notifications Review CloudWatch Automated Response

  CloudTrail AWS config CloudWatch AutoScaling

- **Failure Management**
  - Automation
  - Analyze
  - Backup Data Regularly
  - Recovery and Testing

  CloudFormation S3 Glacier KMS

- **The performance efficiency pillar**

    - The ability to use computing resources efficiently to meet system requirements and to maintain that efficiency as demand changes and technologies evolve

The main Key Service is CloudWatch:

The focus areas include (Key Services for Performance Efficiency:):

- Selection (Optimal server configuration for a particular architecture may vary based on application design)

  EBS Auto Scaling S3 RDS DynamoDB VPC S3 DirecConnect

- Review (Load testing with CloudFormation template in a test environment) Blog and What's New

- Monitoring
    - To monitor and send notification alarms
    - Use Automation to work around performances issues by trigering actions through the following services: CloudWatch Kinesis SQS Lambda

- Trade-offs
  You can trade-off where space (memory or storage) is used to reduce processing time (compute)
  CloudFront AWS DirectConnect ElastiCache SnowBall RDS

Performance Efficiency in the Cloud/AWS Environment

- Go global in minutes
- Use serverless architectures (no need to do setups)
- Experiment more often

The Cost optimization pillar (avoid unneeded costs)

The ability to avoid or eliminate unneeded cost or suboptimal resources

The main service is: Cost allocation Tags

The focus areas include (Key Services for Cost Optimization):

- Cost-effective resources such as taking the right EC2 (On Demand, Reserved Instances, Spot instances)
  Using the appropriate instances and resources for your system is key to cost savings
  Reserved Instances (RIs) Spot Instances Amazon Cost Explorer (Review Costs)

- Match supply and demand to delivers the lowest cost for a system:
    - Provisiong Resources to Match Demand

    - AutoScaling
    - Monitoring tools
    - Benchmarking

- Expenditure (gasto) awareness
  Accurate cost attribution allows you to understand your cost across multiple services, business units, etc.
  The Focus areas include:
    - Tag resources
    - Track project lifecycle and profile applications
    - Monitor usage and spend
    - Cost Explorer (to see for example which services you use more)
    - Partner tools

  Use cost allocation tags and Cost Explorer to categorize and track your AWS costs

- Optimizing over time (taking advantage of new services)
  As AWS releases new service and features, it is a best practice to reassess your existing architectural decisions to ensure they continue to be the most cost effective
  Blog and What's New **Trusted advisor (inspects your AWS environment and finds opportunities to save money by eliminating unused or idle resources or committing to Reserved Instance capacity)**

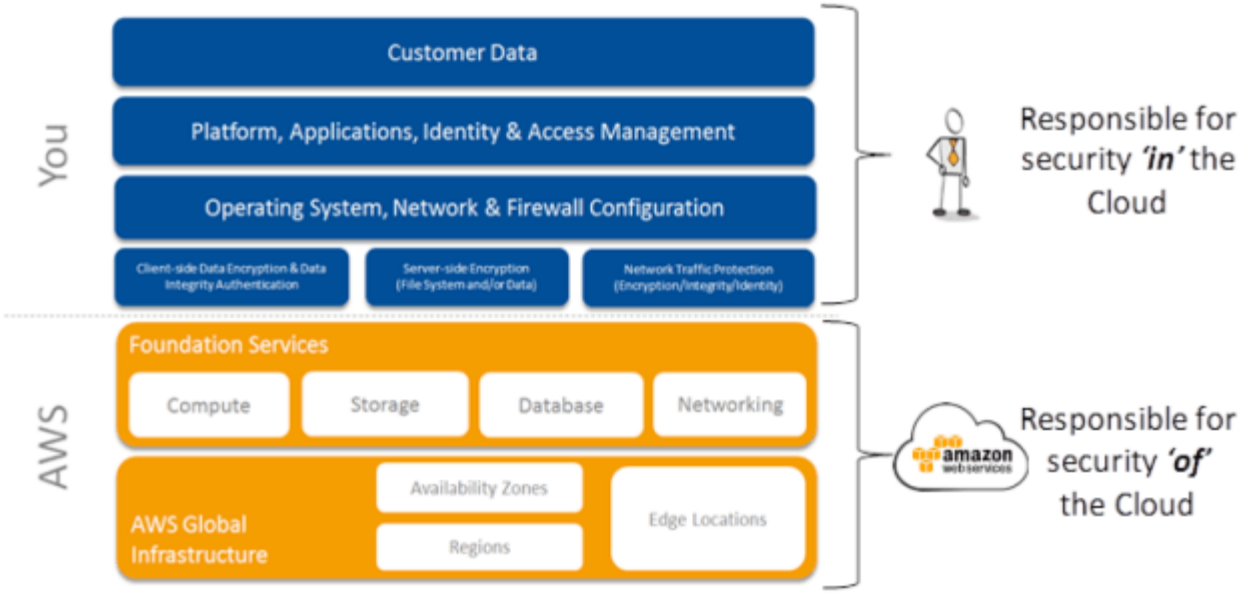
  Probably, AWS has introduced new services since I have written this guide in June 2018. It is good to go through the FAQ of the services. I might have missed few topics, it would be good if you go hrough the exam blueprint and try few sample exam questions 😊

AWS Security - White paper

AWS Security whitepaper is one of the most important whitepaper for the Certification perspective

Shared Security Responsibility Model

- In the Shared Security Responsibility Model, AWS is responsible for securing the underlying infrastructure that supports the cloud, and you're responsible for anything you put on the cloud or connect to the cloud.



AWS Security Responsibilities

- **AWS is responsible for protecting the global infrastructure that runs all of the services offered in the AWS cloud**. This infrastructure is comprised of the hardware, software, networking, and facilities that run AWS services.
- AWS provide several reports from third-party auditors who have verified their compliance with a variety of computer security standards and regulations
- AWS is responsible for the security configuration of its products that are considered managed services for e.g. RDS, DynamoDB
- For Managed Services, AWS will handle basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery.

Customer Security Responsibilities

- AWS Infrastructure as a Service (IaaS) products for e.g. EC2, VPC, S3 are completely under your control and require you to perform all of the necessary security configuration and management tasks.

- Management of the guest OS (including updates and security patches), any application software or utilities installed on the instances, and the configuration of the AWS-provided firewall (called a security group) on each instance

- For most of these managed services, all you have to do is configure logical access controls for the resources and protect the account credentials
- AWS Global Infrastructure Security

AWS Compliance Program

IT infrastructure that AWS provides to its customers is designed and managed in alignment with security best practices and a variety of IT security standards, including:

- SOC 1/SSAE 16/ISAE 3402 (formerly SAS 70)
- SOC 2
- SOC 3
- FISMA, DIACAP, and FedRAMP
- DOD CSM Levels 1-5
- PCI DSS Level 1
- ISO 9001 / ISO 27001
- ITAR
- FIPS 140-2
- MTCS Level 3

And meet several industry-specific standards, including:

- Criminal Justice Information Services (CJIS)
- Cloud Security Alliance (CSA)
- Family Educational Rights and Privacy Act (FERPA)
- Health Insurance Portability and Accountability Act (HIPAA)
- Motion Picture Association of America (MPAA)
  - Physical and Environmental Security
  - Storage Decommissioning
- When a storage device has reached the end of its useful life, AWS procedures include a decommissioning process that is designed to prevent customer data from being exposed to unauthorized individuals.
- AWS uses the techniques detailed in DoD 5220.22-M (National Industrial Security Program Operating Manual) or NIST 800-88 (Guidelines for Media Sanitization) to destroy data as part of the decommissioning process.
- All decommissioned magnetic storage devices are degaussed and physically destroyed in accordance with industry-standard practices.

- Network Security

Amazon Corporate Segregation

- AWS Production network is segregated from the Amazon Corporate network and requires a separate set of credentials for logical access.

- Amazon Corporate network relies on user IDs, passwords, and Kerberos, while the AWS Production network require SSH public-key authentication through a bastion host.

Networking Monitoring & Protection

AWS utilizes a wide variety of automated monitoring systems to provide a high level of service performance and availability. These tools monitor server and network usage, port scanning activities, application usage, and unauthorized intrusion attempts. The tools have the ability to set custom performance metrics thresholds for unusual activity.

AWS network provides protection against traditional network security issues

1. **DDOS** – AWS uses proprietary DDoS mitigation techniques. Additionally, AWS's networks are multi-homed across a number of providers to achieve Internet access diversity.

2. **Man in the Middle attacks** – AWS APIs are available via SSL-protected endpoints which provide server authentication

3. **IP spoofing** – AWS-controlled, host-based firewall infrastructure will not permit an instance to send traffic with a source IP or MAC address other than its own.

4. **Port Scanning** – Unauthorized port scans by Amazon EC2 customers are a violation of the AWS Acceptable Use Policy. When unauthorized port scanning is detected by AWS, it is stopped and blocked. Penetration/Vulnerability testing can be performed only on your own instances, with mandatory prior approval, and must not violate the AWS Acceptable Use Policy.

5. **Packet Sniffing by other tenants** – It is not possible for a virtual instance running in promiscuous mode to receive or "sniff" traffic that is intended for a different virtual instance. While you can place your interfaces into promiscuous mode, the hyper-visor will not deliver any traffic to them that is not addressed to them. Even two virtual instances that are owned by the same customer located on the same physical host cannot listen to each other's traffic.

Secure Design Principles

AWS's development process follows:

- Secure software development best practices, which include formal design reviews by the AWS Security Team, threat modeling, and completion of a risk assessment
- Static code analysis tools are run as a part of the standard build process
- Recurring penetration testing performed by carefully selected industry experts

AWS Account Security Features

AWS account security features includes credentials for access control, HTTPS endpoints for encrypted data transmission, the creation of separate IAM user accounts, user activity logging for security monitoring, and Trusted Advisor security checks

AWS Credentials

Credential Type	Use	Description
Passwords	AWS root account or IAM user account login to the AWS Management Console	A string of characters used to log into your AWS account or IAM account. AWS passwords must be a minimum of 6 characters and may be up to 128 characters.
Multi-Factor Authentication (MFA)	AWS root account or IAM user account login to the AWS Management Console	A six-digit single-use code that is required in addition to your password to log in to your AWS Account or IAM user account.
Access Keys	Digitally signed requests to AWS APIs (using the AWS SDK, CLI, or REST/Query APIs)	Includes an access key ID and a secret access key. You use access keys to digitally sign programmatic requests that you make to AWS.
Key Pairs	• SSH login to EC2 instances • CloudFront signed URLs	A key pair is required to connect to an EC2 instance launched from a public AMI. The keys that Amazon EC2 uses are 1024-bit SSH-2 RSA keys. You can have a key pair generated automatically for you when you launch the instance or you can upload your own.
X.509 Certificates	• Digitally signed SOAP requests to AWS APIs • SSL server certificates for HTTPS	X.509 certificates are only used to sign SOAP-based requests (currently used only with Amazon S3). You can have AWS create an X.509 certificate and private key that you can download, or you can upload your own certificate by using the Security Credentials page.

Individual User Accounts
Do not use the Root account, instead create an IAM User for each user and provide them with a unique set of Credentials and grant least privilege as required to perform their job function

Secure HTTPS Access Points
Use HTTPS, provided by all AWS services, for data transmissions, which uses public-key cryptography to prevent eavesdropping, tampering, and forgery

Security Logs
Use Amazon CloudTrail which provides logs of all requests for AWS resources within the account and captures information about every API call to every AWS resource you use, including sign-in events

Trusted Advisor Security Checks
Use Trusted Advisor service which helps inspect AWS environment and provide recommendations when opportunities may exist to optimize cost, improve system performance, or close security gaps

AWS DDoS Resiliency – Best Practices – Whitepaper

- **Denial of Service (DoS) is an attack, carried out by a single attacker, which attempts to make a website or application unavailable to the end users.**

- Distributed Denial of Service (DDoS) is an attack, carried out by multiple attackers either controlled or compromised by a group of collaborators, **which generates a flood of requests to the application making in unavailable to the legitimate end users**

Mitigation techniques

Minimize the Attack Surface Area

- This is all all about reducing the attack surface, the different Internet entry points, that allows access to your application
-
- **Strategy to minimize the Attack surface area:**
    - reduce the number of necessary Internet entry points,
    - don’t expose back end servers,
    - eliminate non-critical Internet entry points,
    - separate end user traffic from management traffic,
    - obfuscate necessary Internet entry points to the level that untrusted end users cannot access them, and
    - decouple Internet entry points to minimize the effects of attacks.

- **Benefits**
  - Minimizes the effective attack vectors and targets
  - Less to monitor and protect

- **Strategy can be achieved using AWS Virtual Private Cloud (VPC)**
  - helps define a logically isolated virtual network within the AWS
  - provides ability to create Public & Private Subnets to launch the Internet facing and non-public facing instances accordingly
  - **provides NAT gateway which allows instances in the private subnet to have internet access without the need to launch them in public subnets with Public IPs**
  - **allows creation of Bastion host which can be used to connect to instances in the private subnets**
  - provides the ability to configure security groups for instances and NACLs for subnets, which act as a firewall, to control and limit outbound and inbound traffic

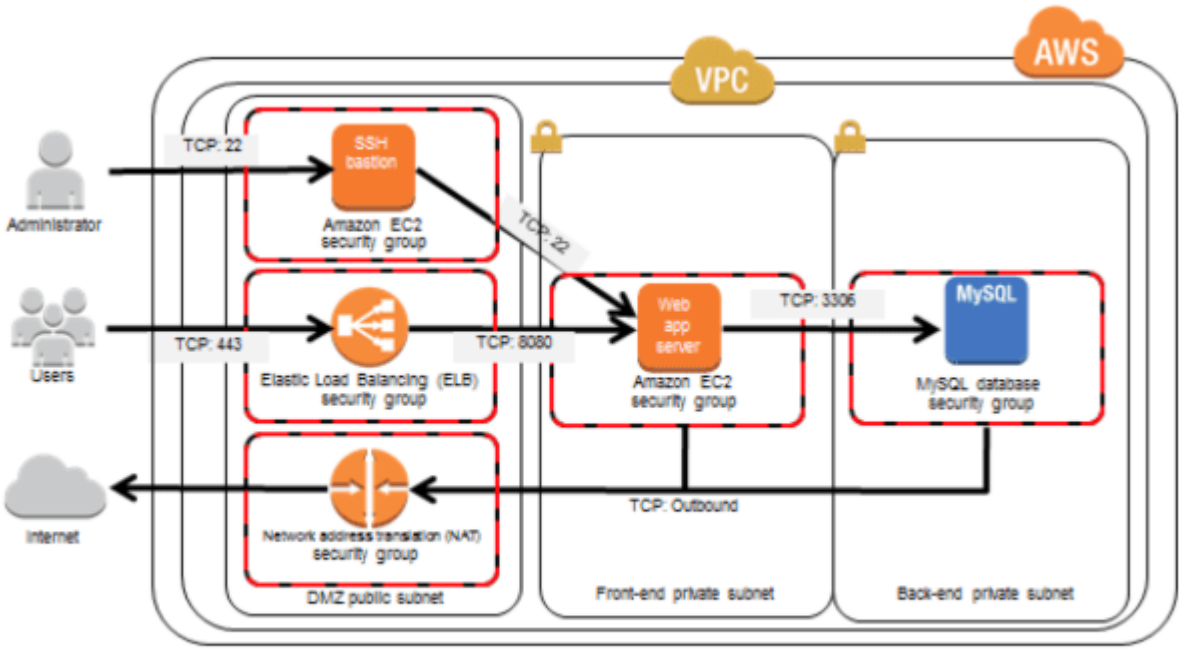


Figure 2. Reference architecture with Amazon VPC configuration

Be Ready to Scale to Absorb the Attack

- DDOS mainly targets to load the systems till the point they cannot handle the load and are rendered unusable.

- **Scaling out Benefits**
  - help build a resilient architecture
  - makes the attacker work harder
  - gives you time to think, analyze and adapt

- **AWS provided services :-**
  - **Auto Scaling & ELB**
    - Horizontal scaling using Auto Scaling with ELB

- Auto Scaling allows instances to be added and removed as the demand changes
- ELB helps distribute the traffic across multiple EC2 instances while acting as a Single point of contact.
- Auto Scaling automatically registers and de-registers EC2 instances with the ELB during scale out and scale in events

- **EC2 Instance**
  - Vertical scaling can be achieved by using appropriate EC2 instance types for e.g. EBS optimized or ones with 10 gigabyte network connectivity to handle the load

- **Enhanced Networking**
  - Use Instances with Enhanced Networking capabilities which can provide high packet-per-second performance, low latency networking, and improved scalability

- **Amazon CloudFront**
  - CloudFront is a CDN, acts as a proxy between end users and the Origin servers, and helps distribute content to the end users without sending traffic to the Origin servers.

  - CloudFront has the inherent ability to help mitigate against both infrastructure and some application layer DDoS attacks by dispersing the traffic across multiple locations.

  - AWS has multiple Internet connections for capacity and redundancy at each location, which allows it to isolate attack traffic while serving content to legitimate end users

  - CloudFront also has filtering capabilities to ensure that only valid TCP connections and HTTP requests are made while dropping invalid requests. This takes the burden of handling invalid traffic (commonly used in UDP & SYN floods, and slow reads) off the origin.

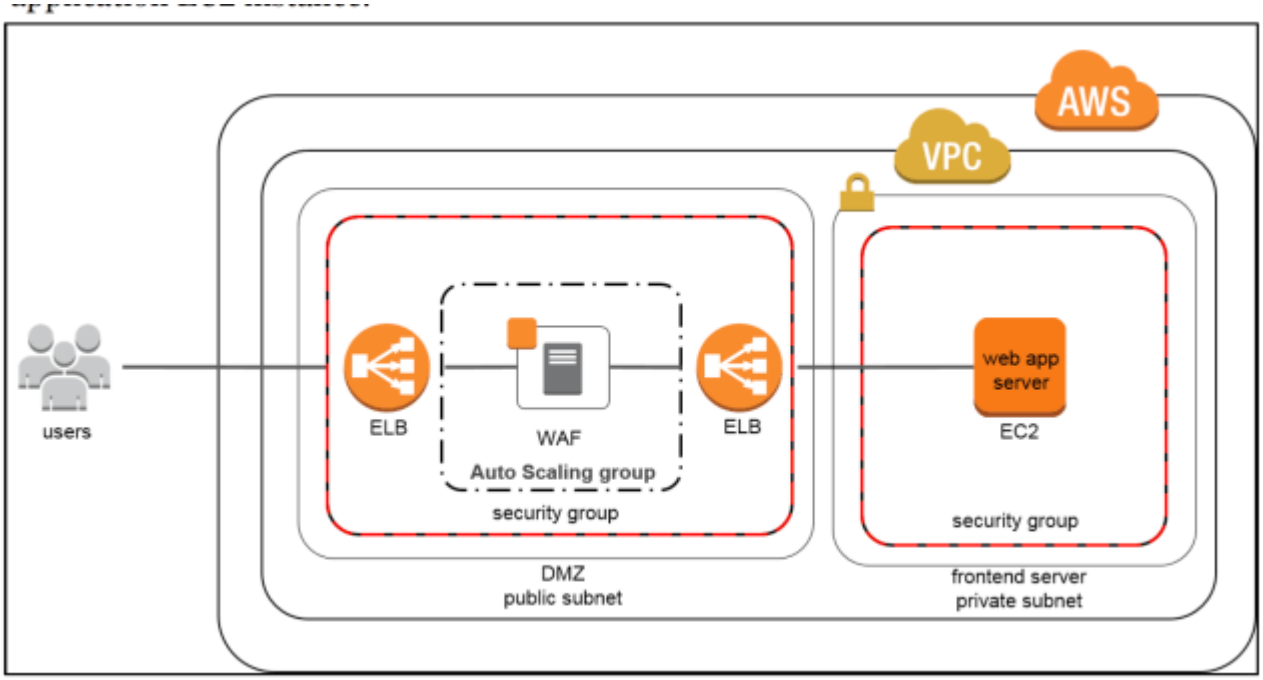
- **Route 53**
  - DDOS attacks are also targeted towards DNS, cause if the DNS is unavailable your application is effectively unavailable.

  - AWS Route 53 is highly available and scalable DNS service and have capabilities to ensure access to the application even when under DDOS attack
    - Shuffle Sharding – Shuffle sharding is similar to the concept of database sharding, where horizontal partitions of data are spread across separate database servers to spread load and provide redundancy. Similarly, Amazon Route 53 uses shuffle sharding to spread DNS requests over numerous PoPs, thus providing multiple paths and routes for your application.

- Anycast Routing – Anycast routing increases redundancy by advertising the same IP address from multiple PoPs. In the event that a DDoS attack overwhelms one endpoint, shuffle sharding isolate failures while providing additional routes to your infrastructure.

Safeguard Exposed & Hard to Scale Expensive Resources

- If entry points cannot be limited, additional measures to restrict access and protect those entry points without interrupting legitimate end user traffic
-
- **AWS provided services :-**
  - CloudFront
    - CloudFront can restrict access to content using Geo Restriction and Origin Access Identity
    - With Geo Restriction, access can be restricted to a set of whitelisted countries or prevent access from a set of black listed countries
    - Origin Access Identity is the CloudFront special user which allows access to the resources only through CloudFront while denying direct access to the origin content for e.g. if S3 is the Origin for CloudFront, S3 can be configured to allow access only from OAI and hence deny direct access
  - Route 53
    - Route 53 provides two features Alias Record sets & Private DNS to make it easier to scale infrastructure and respond to DDoS attacks
  - **WAF**
    - **WAFs act as filters that apply a set of rules to web traffic. Generally, these rules cover exploits like cross-site scripting (XSS) and SQL injection (SQLi) but can also help build resiliency against DDoS by mitigating HTTP GET or POST** floods
    - WAF provides a lot of features like
      - OWASP Top 10
      - HTTP rate limiting (where only a certain number of requests are allowed per user in a timeframe),
      - Whitelist or blacklist (customizable rules)
      - inspect and identify requests with abnormal patterns,
      - CAPTCHA etc
    - To prevent WAF from being a Single point of failure, a WAF sandwich pattern can be implemented where an auto scaled WAF sits between the Internet and Internal Load Balancer



Learn Normal Behavior

- Understand the normal usual levels and Patterns of traffic for your application and use that as a benchmark for identifying abnormal level of traffic or resource spikes patterns

- Benefits
    - allows one to spot abnormalities
    - configure Alarms with accurate thresholds
    - assists with generating forensic data

- **AWS provided services for tracking**

    - **AWS CloudWatch monitoring**
        - CloudWatch can be used to monitor your infrastructure and applications running in AWS. Amazon CloudWatch can collect metrics, log files, and set alarms for when these metrics have passed predetermined thresholds

    - **VPC Flow Logs**
        - Flow logs helps capture traffic to the Instances in an VPC and can be used to understand the pattern

Create a Plan for Attacks

- Have a plan in place before an attack, which ensures that:
    - Architecture has been validated and techniques selected work for the infrastructure
    - Costs for increased resiliency have been evaluated and the goals of your defense are understood
    - Contact points have been identified

AWS Disaster Recovery Whitepaper

- AWS Disaster Recovery whitepaper highlights AWS services and features that can be leveraged for **disaster recovery (DR)** processes to significantly minimize the impact on data, system, and overall business operations.

- It outlines best practices to improve your DR processes, from minimal investments to full-scale availability and fault tolerance, and describes how AWS services can be used to reduce cost and ensure business continuity during a DR event

- **Disaster recovery (DR) is about preparing for and recovering from a disaster**. Any event that has a negative impact on a company's business continuity or finances could be termed a disaster. One of the AWS best practice is to always design your systems for failures

Disaster Recovery Key AWS services

1. **Region**
    - AWS services are available in multiple regions around the globe, and the DR site location can be selected as appropriate, in addition to the primary site location

2. **Storage**
    - **Amazon S3**
        - provides a highly durable (99.999999999%) storage infrastructure designed for mission-critical and primary data storage.
        - stores Objects redundantly on multiple devices across multiple facilities within a region

    - **Amazon Glacier**
        - provides extremely low-cost storage for data archiving and backup.
        - Objects are optimized for infrequent access, for which retrieval times of several (3-5) hours are adequate.

    - **Amazon EBS**
        - provides the ability to create point-in-time snapshots of data volumes.
        - Snapshots can then be used to create volumes and attached to running instances

    - **Amazon Storage Gateway**
        - a service that provides seamless and highly secure integration between on-premises IT environment and the storage infrastructure of AWS.

- **AWS Import/Export**
  - accelerates moving large amounts of data into and out of AWS by using portable storage devices for transport bypassing the Internet
  - transfers data directly onto and off of storage devices by means of the high-speed internal network of Amazon

3. **Compute**

- **Amazon EC2**
  - provides resizeable compute capacity in the cloud which can be easily created and scaled.
  - EC2 instance creation using Preconfigured AMIs
  - EC2 instances can be launched in multiple AZs, which are engineered to be insulated from failures in other AZs

4. **Networking**

- **Amazon Route 53**
  - is a highly available and scalable DNS web service
  - includes a number of global load-balancing capabilities that can be effective when dealing with DR scenarios for e.g. DNS endpoint health checks and the ability to failover between multiple endpoints

- **Elastic IP**
  - addresses enables masking of instance or Availability Zone failures by programmatically remapping
  - addresses are static IP addresses designed for dynamic cloud computing.

- **Elastic Load Balancing (ELB)**
  - performs health checks and automatically distributes incoming application traffic across multiple EC2 instances

- **Amazon Virtual Private Cloud (Amazon VPC)**
  - allows provisioning of a private, isolated section of the AWS cloud where resources can be launched in a defined virtual network

- **Amazon Direct Connect**
  - makes it easy to set up a dedicated network connection from on-premises environment to AWS

5. **Databases**

- **RDS, DynamoDb, Redshift** provided as a fully managed RDBMS, NoSQL and data warehouse solutions which can scale up easily
- DynamoDB offers cross region replication
- RDS provides Multi-AZ and Read Replicas and also ability to snapshot data from one region to other

6. **Deployment Orchestration**
    - **CloudFormation**
        - gives developers and systems administrators an easy way to create a collection of related AWS resources and provision them in an orderly and predictable fashion

    - **Elastic Beanstalk**
        - is an easy-to-use service for deploying and scaling web applications and services

    - **OpsWorks**
        - is an application management service that makes it easy to deploy and operate applications of all types and sizes.
        - Environment can be defined as a series of layers, and each layer can be configured as a tier of the application.
        - has automatic host replacement, so in the event of an instance failure it will be automatically replaced.
        - can be used in the preparation phase to template the environment, and combined with AWS CloudFormation in the recovery phase.
        - Stacks can be quickly provisioned from the stored configuration to support the defined RTO.

Key factors for Disaster Planning

A DR plan illustrating the chronology of the RPO and the RTO with respect to the MI.

- **Recovery Time Objective (RTO) –** The time it takes after a disruption to restore a business process to its service level, as defined by the operational level agreement (OLA) for e.g. if the RTO is 1 hour and disaster occurs @ 12:00 p.m (noon), then the DR process should restore the systems to an acceptable service level within an hour i.e. by 1:00 p.m
- **Recovery Point Objective (RPO)** – The acceptable amount of data loss measured in time before the disaster occurs. for e.g., if a disaster occurs at 12:00 p.m (noon) and the RPO is one hour, the system should recover all data that was in the system before 11:00 a.m.

Disaster Recovery Scenarios

- Disaster Recovery scenarios can be implemented with the Primary infrastructure running in your data center in conjunction with the AWS
- Disaster Recovery Scenarios still apply if Primary site is running in AWS using AWS multi region feature.
- Combination and variation of the below is always possible.

Disaster Recovery Scenarios Options

1. Backup & Restore (Data backed up and restored)
2. Pilot Light (Only Minimal critical functionalities)
3. Warm Standby (Fully Functional Scaled down version)
4. Multi-Site (Active-Active)

For the DR scenarios options, RTO and RPO reduces with an increase in Cost as you move from Backup & Restore option (left) to Multi-Site option (right)

Backup & Restore

AWS can be used to backup the data in a cost effective, durable and secure manner as well as recover the data quickly and reliably.

Backup phase

In most traditional environments, data is backed up to tape and sent off-site regularly taking longer time to restore the system in the event of a disruption or disaster

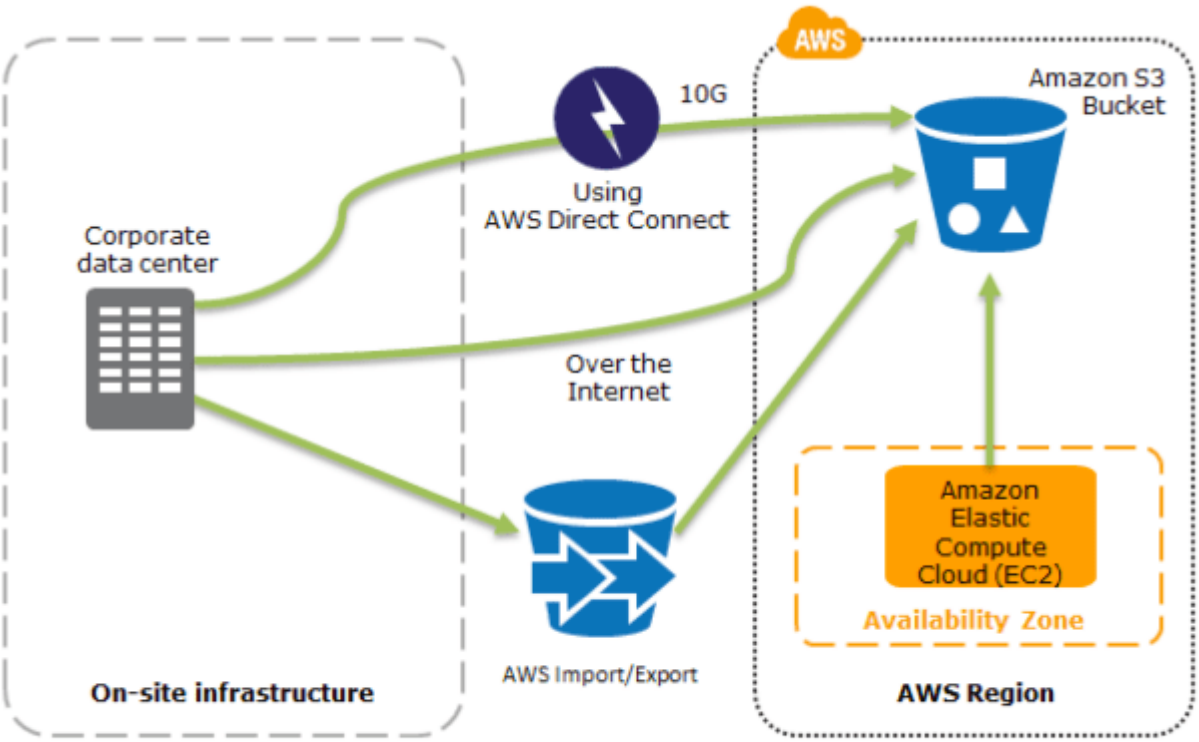


Figure 2: Data Backup Options to Amazon S3 from On-Site Infrastructure or from AWS.

1. Amazon S3 can be used to backup the data and perform a quick restore and is also available from any location
2. AWS Import/Export can be used to transfer large data sets by shipping storage devices directly to AWS bypassing the Internet
3. Amazon Glacier can be used for archiving data, where retrieval time of several hours are adequate and acceptable

4. AWS Storage Gateway enables snapshots (used to created EBS volumes) of the on-premises data volumes to be transparently copied into S3 for backup. It can be used either as a backup solution (Gateway-stored volumes) or as a primary data store (Gateway-cached volumes)
5. AWS Direct connect can be used to transfer data directly from On-Premise to Amazon consistently and at high speed
6. Snapshots of Amazon EBS volumes, Amazon RDS databases, and Amazon Redshift data warehouses can be stored in Amazon S3
7.

Restore phase

Data backed up then can be used to quickly restore and create Compute and Database instances

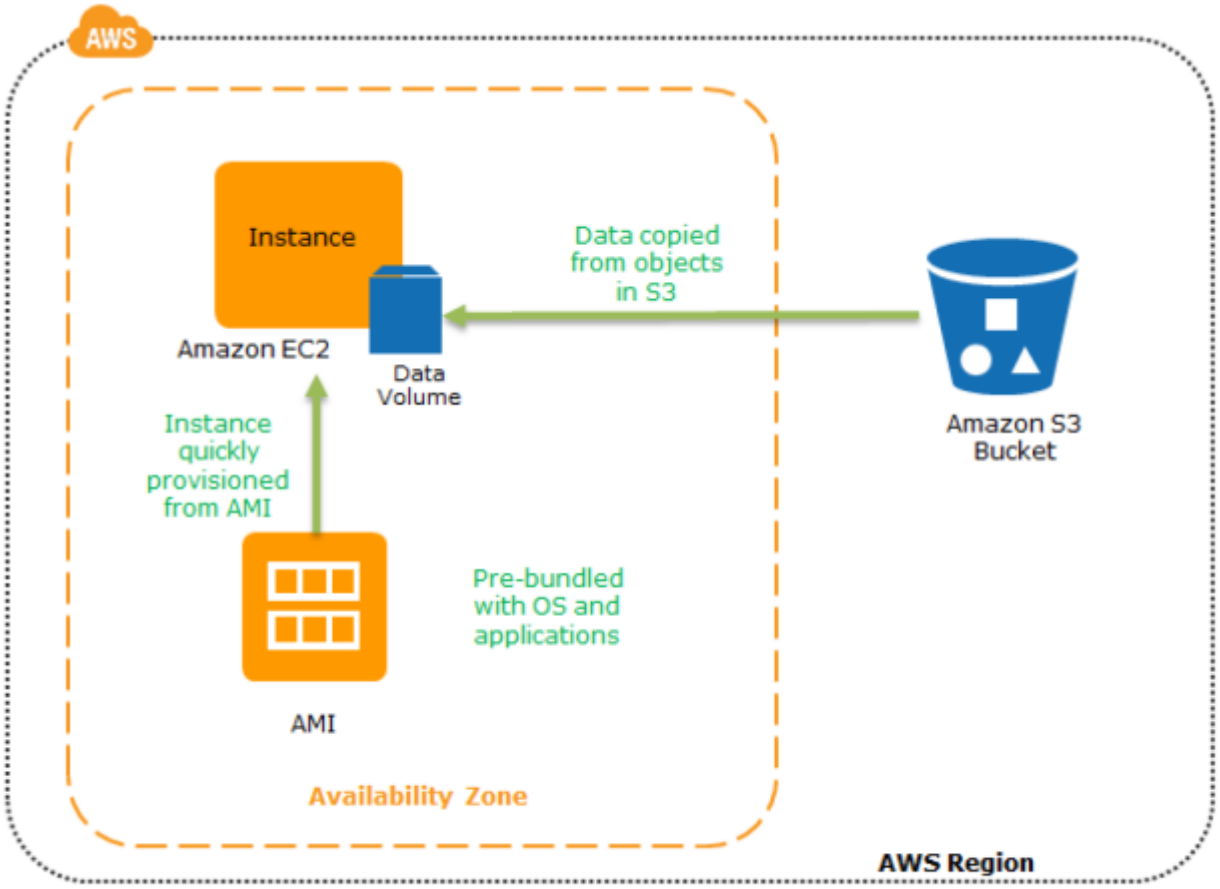


Figure 3: Restoring a System from Amazon S3 Backups to Amazon EC2

Key steps for Backup and Restore:
1. Select an appropriate tool or method to back up the data into AWS.
2. Ensure an appropriate retention policy for this data.
3. Ensure appropriate security measures are in place for this data, including encryption and access policies.
4. Regularly test the recovery of this data and the restoration of the system.

Pilot Light

In a Pilot Light Disaster Recovery scenario option a minimal version of an environment is always running in the cloud, which basically host the critical functionalities of the application for e.g. databases
In this approach :-

1. Maintain a pilot light by configuring and running the most critical core elements of your system in AWS for e.g. Databases where the data needs to be replicated and kept updated.
2. During recovery, a full-scale production environment, for e.g. application and web servers, can be rapidly provisioned (using preconfigured AMIs and EBS volume snapshots) around the critical core
3. For Networking, either a ELB to distribute traffic to multiple instances and have DNS point to the load balancer or preallocated Elastic IP address with instances associated can be used

Preparation phase steps :

1. Set up Amazon EC2 instances or RDS instances to replicate or mirror data critical data
2. Ensure that all supporting custom software packages available in AWS.
3. Create and maintain AMIs of key servers where fast recovery is required.
4. Regularly run these servers, test them, and apply any software updates and configuration changes.
5. Consider automating the provisioning of AWS resources.

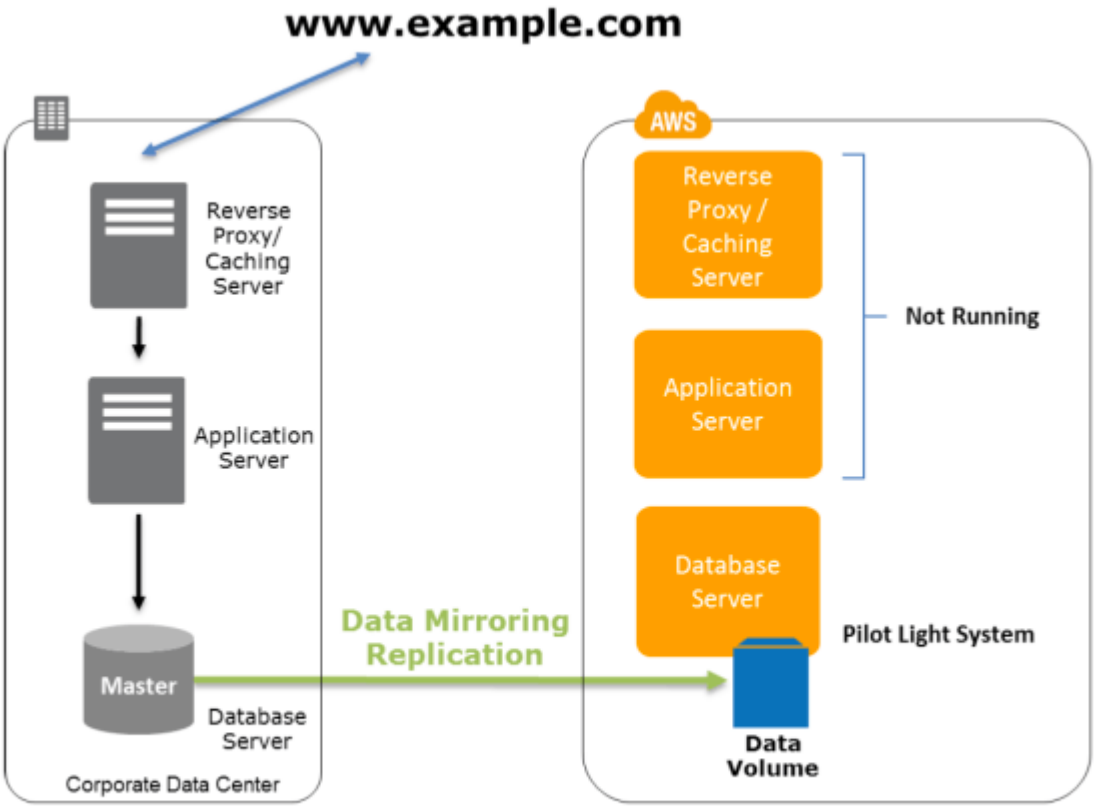


Figure 4: The Preparation Phase of the Pilot Light Scenario

Recovery Phase steps :

1. Start the application EC2 instances from your custom AMIs.

2. Resize existing database/data store instances to process the increased traffic for e.g. If using RDS, it can be easily scaled vertically while EC2 instances can be easily scaled horizontally
3. Add additional database/data store instances to give the DR site resilience in the data tier for e.g. turn on Multi-AZ for RDS to improve resilience.
4. Change DNS to point at the Amazon EC2 servers.
5. Install and configure any non-AMI based systems, ideally in an automated way.
6.

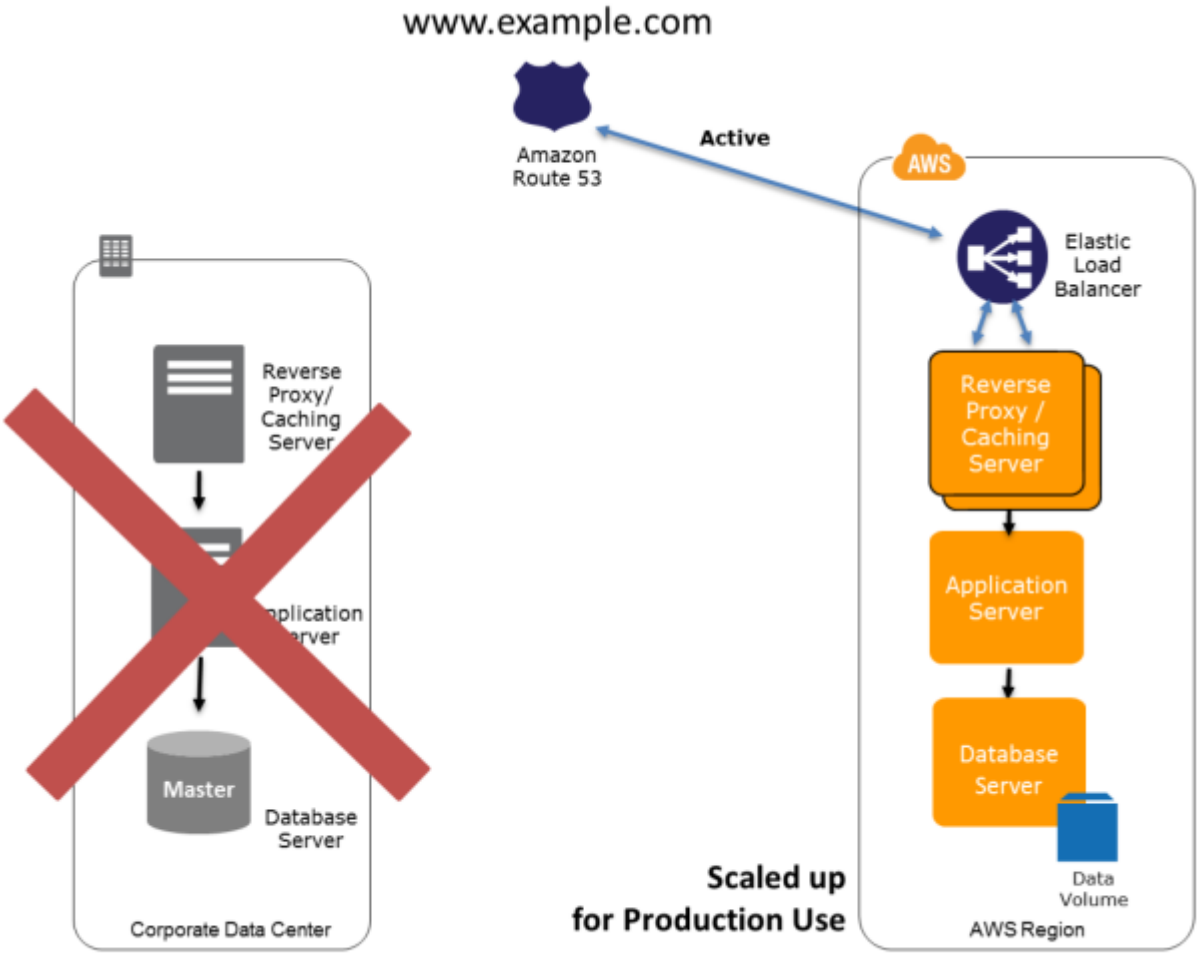


Figure 5: The Recovery Phase of the Pilot Light Scenario.

Warm Standby

- In a Warm standby DR scenario a scaled-down version of a fully functional environment identical to the business critical systems is always running in the cloud
- This setup can be used for testing, quality assurances or for internal use.
- In case of an disaster, the system can be easily scaled up or out to handle production load.

Preparation phase steps :

1. Set up Amazon EC2 instances to replicate or mirror data.
2. Create and maintain AMIs for faster provisioning
3. Run the application using a minimal footprint of EC2 instances or AWS infrastructure.

4. Patch and update software and configuration files in line with your live environment.

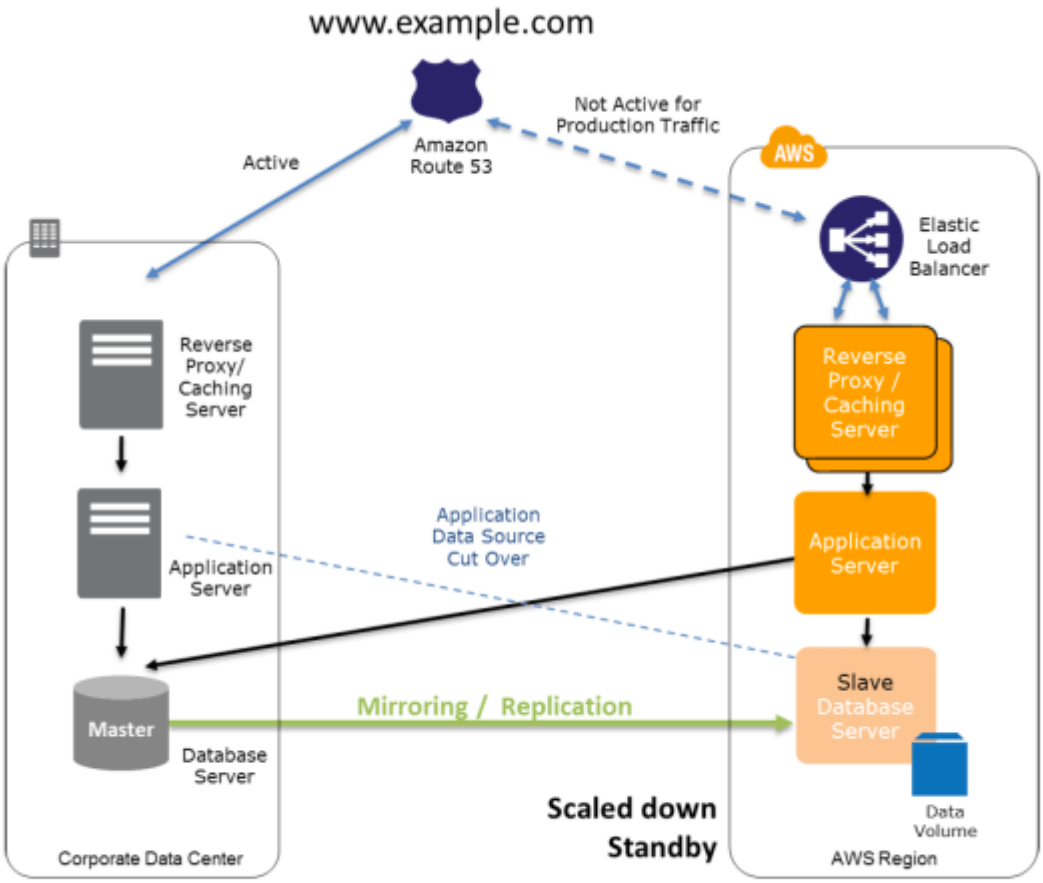


Figure 6: The Preparation Phase of the Warm Standby Scenario.

Recovery phase Steps:

1. Increase the size of the Amazon EC2 fleets in service with the load balancer (horizontal scaling).
2. Start applications on larger Amazon EC2 instance types as needed (vertical scaling).
3. Either manually change the DNS records, or use Route 53 automated health checks to route all the traffic to the AWS environment.
4. Consider using Auto Scaling to right-size the fleet or accommodate the increased load.
5. Add resilience or scale up your database to guard against DR going down

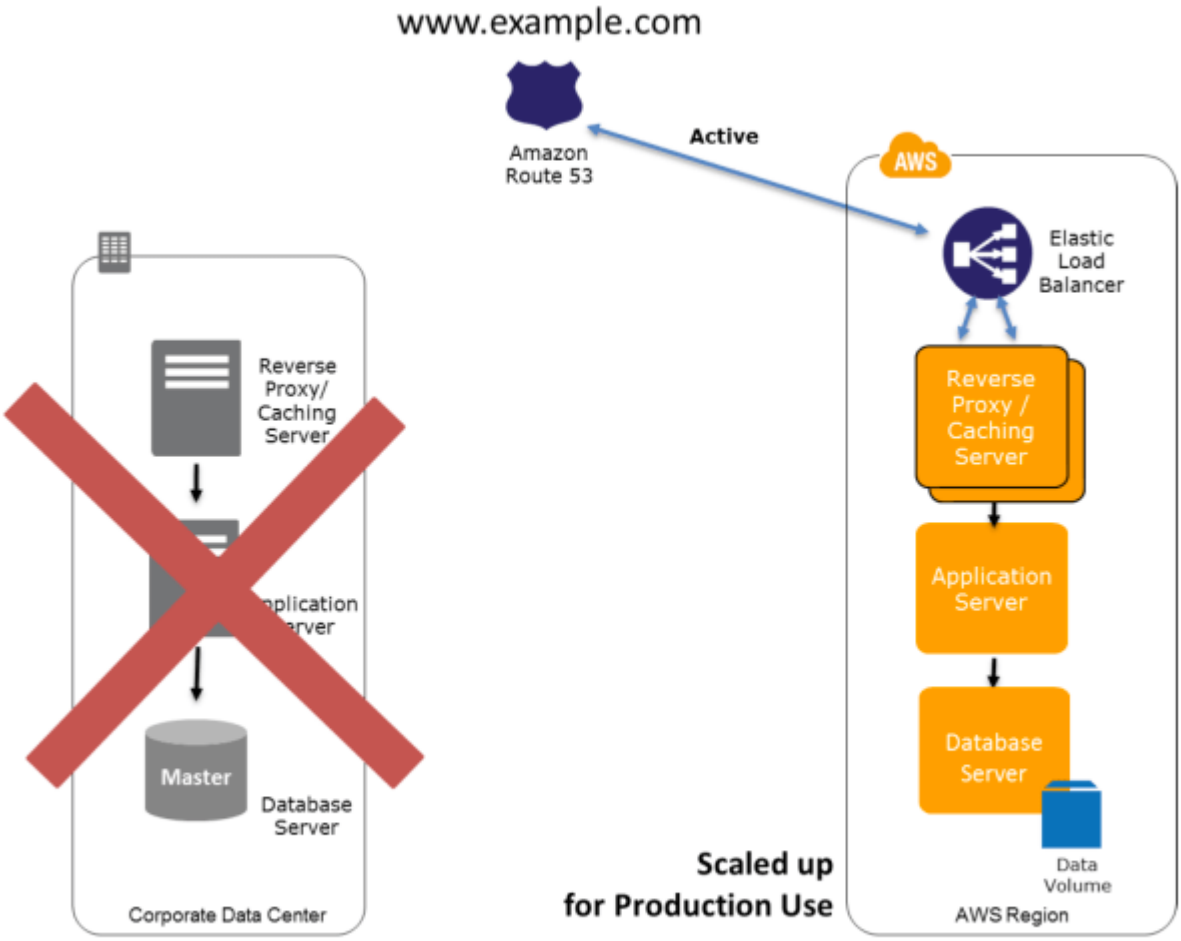


Figure 7: The Recovery Phase of the Warm Standby Scenario.

Multi-Site

- Multi-Site is an active-active configuration DR approach, where in an identical solution runs on AWS as your on-site infrastructure.
- Traffic can be equally distributed to both the infrastructure as needed by using DNS service weighted routing approach.
- In case of a disaster the DNS can be tuned to send all the traffic to the AWS environment and the AWS infrastructure scaled accordingly.

Preparation phase steps :

1. Set up your AWS environment to duplicate the production environment.
2. Set up DNS weighting, or similar traffic routing technology, to distribute incoming requests to both sites.
3. Configure automated failover to re-route traffic away from the affected site. for e.g. application to check if primary DB is available if not then redirect to the AWS DB

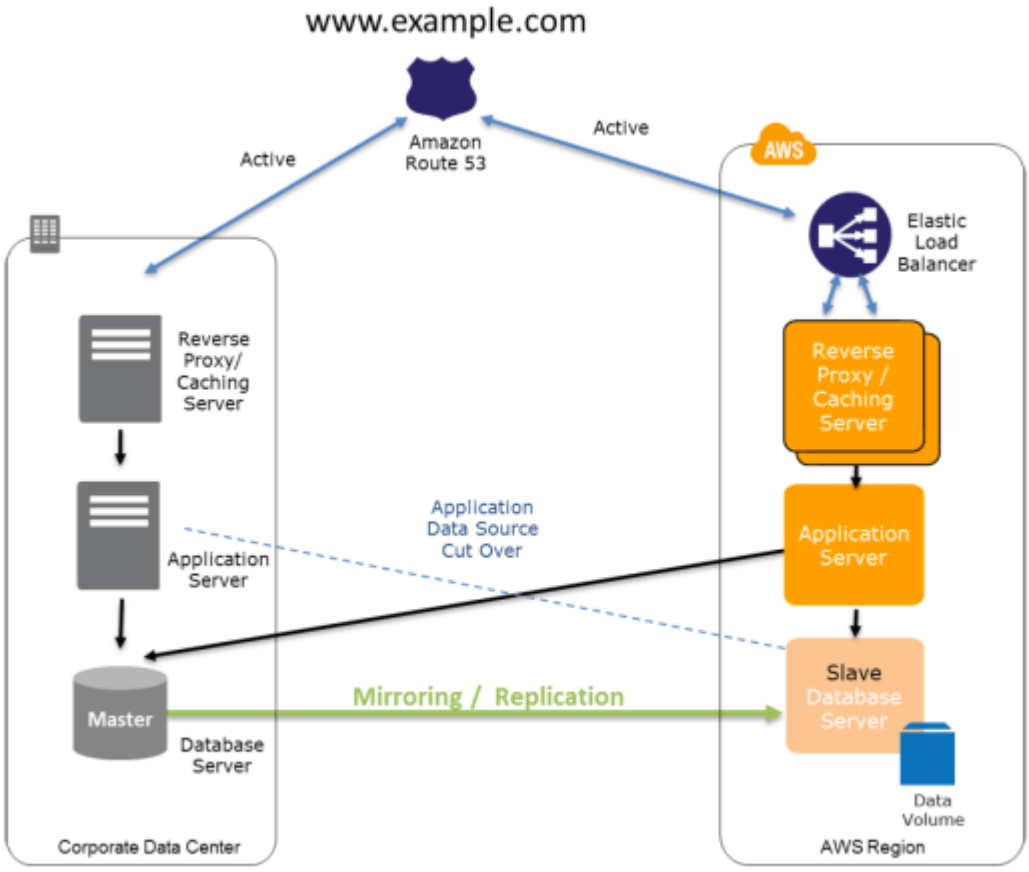


Figure 8: The Preparation Phase of the Multi-Site Scenario.

Recovery phase steps :

1. Either manually or by using DNS failover, change the DNS weighting so that all requests are sent to the AWS site.
2. Have application logic for failover to use the local AWS database servers for all queries.

3. Consider using Auto Scaling to automatically right-size the AWS fleet.

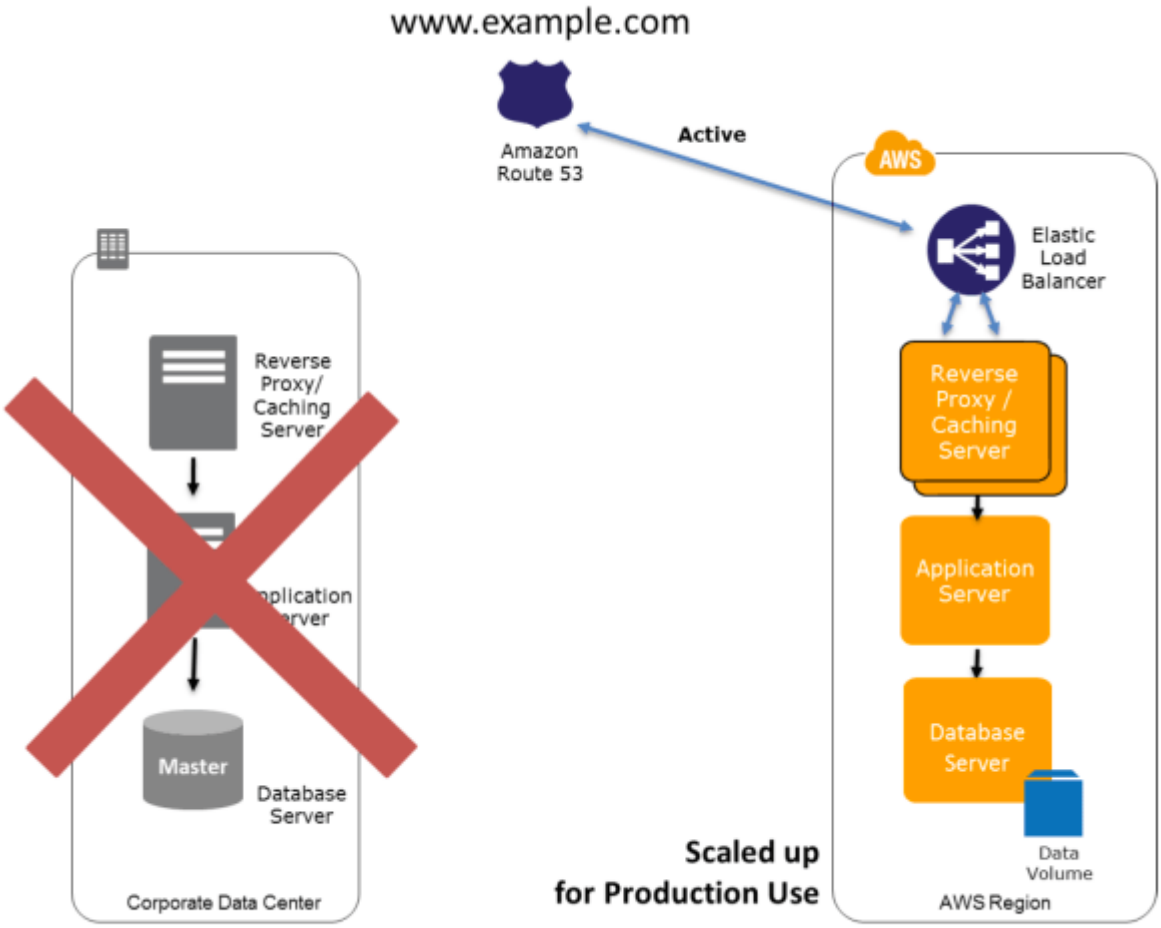


Figure 9: The Recovery Phase of the Multi-Site Scenario Involving On-Site and AWS Infrastructure.

AWS Best Practices – Whitepaper (Interesting to Read)

Architecting for the Cloud – AWS Best Practices whitepaper provides architectural patterns and advice on how to design systems that are secure, reliable, high performing, and cost efficient

AWS Design Principles

Scalability

- While AWS provides virtually unlimited on-demand capacity, the architecture should be designed to take advantage of those resources
-
- **There are two ways to scale an IT architecture**
    - Vertical Scaling
        - takes place through increasing specifications of an individual resource for e.g. updating EC2 instance type with increasing RAM, CPU, IOPS, or networking capabilities
        - will eventually hit a limit, and is not always a cost effective or highly available approach
    - Horizontal Scaling
        - takes place through increasing number of resources for e.g. adding more EC2 instances or EBS volumes
        - can help leverage the elasticity of cloud computing
        - not all the architectures can be designed to distribute their workload to multiple resources
        - applications designed should be stateless,
            - that needs **no knowledge of previous interactions and stores no session information**
            - capacity can be increased and decreased, after running tasks have been drained
        - State, if needed, can be implemented using
            - Low latency external store, for e.g. DynamoDB, Redis, to maintain state information
            - Session affinity, for e.g. ELB sticky sessions, to bind all the transactions of a session to a specific compute resource. However, it cannot be guaranteed or take advantage of newly added resources for existing sessions
        - Load can be distributed across multiple resources using
            - Push model, for e.g. through ELB where it distributes the load across multiple EC2 instances
            - Pull model, for e.g. through SQS or Kinesis where multiple consumers subscribe and consume
        - Distributed processing, for e.g. using EMR or Kinesis, helps process large amounts of data by dividing task and its data into many small fragments of works

• Disposable Resources Instead of Fixed Servers

- Resources need to be treated as temporary disposable resources rather then fixed permanent on-premises resources before

- **AWS focuses on the concept of Immutable infrastructure**
  - servers once launched, is never updated throughout its lifetime.
  - updates can be performed on a new server with latest configurations,
  - this ensures resources are always in a consistent (and tested) state and easier roll-backs

- **AWS provides multiple ways to instantiate compute resources in an automated and repeatable way:**

  - Bootstraping
    - scripts to configure and setup for e.g. using data scripts and cloud-init to install software or copy resources and code

  - Golden Images
    - a snapshot of a particular state of that resource,
    - faster start times and removes dependencies to configuration services or third-party repositories

  - Containers
    - AWS support for docker images through Elastic Beanstalk and ECS
    - Docker allows packaging a piece of software in a Docker Image, which is a standardized unit for software development, containing everything the software needs to run: code, runtime, system tools, system libraries, etc

- **Infrastructure as Code**
  - AWS assets are programmable, techniques, practices, and tools from software development can be applied to make the whole infrastructure reusable, maintainable, extensible, and testable.
  - AWS provides services like CloudFormation, OpsWorks for deployment

Automation

- AWS provides various automation tools and services which help improve system's stability, efficiency and time to market.
    - **Elastic Beanstalk**
        - a PaaS that allows quick application deployment while handling resource provisioning, load balancing, auto scaling, monitoring etc
    - **EC2 Auto Recovery**
        - creates CloudWatch alarm that monitors an EC2 instance and automatically recovers it if it becomes impaired.
        - A recovered instance is identical to the original instance, including the instance ID, private & Elastic IP addresses, and all instance metadata.
        - Instance is migrated through reboot, in memory contents are lost.
    - **Auto Scaling**
        - allows maintain application availability and scale the capacity up or down automatically as per defined conditions
    - **CloudWatch Alarms**
        - allows SNS triggers to be configured when a particular metric goes beyond a specified threshold for a specified number of periods
    - **CloudWatch Events**
        - allows real-time stream of system events that describe changes in AWS resources
    - **OpsWorks**
        - allows continuous configuration through life-cycle events that automatically update the instances' configuration to adapt to environment changes.
        - Events can be used to trigger Chef/Puppet recipes on each instance to perform specific configuration tasks
    - **Lambda Scheduled Events**
        - allows Lambda function creation and direct AWS Lambda to execute it on a regular schedule.

Loose Coupling

- AWS helps loose coupled architecture that reduces interdependencies, a change or failure in a component does not cascade to other components

    - **Asynchronous Integration**
        - does not involve direct point-to-point interaction but usually through an intermediate durable storage layer for e.g. SQS, Kinesis

        - **decouples the components and introduces additional resiliency**

        - suitable for any interaction that doesn't need an immediate response and where an ack that a request has been registered will suffice

    - **Service Discovery**
        - allows new resources to be launched or terminated at any point in time and discovered as well for e.g. using ELB as a single point of contact with hiding the underlying instance details or Route 53 zones to abstract load balancer's endpoint

    - **Well-Defined Interfaces**
        - allows various components to interact with each other through specific, technology agnostic interfaces for e.g. RESTful apis with API Gateway

Services, Not Servers

Databases

- AWS provides different categories of database technologies:

    - **Relational Databases (RDS)**
        - normalizes data into well-defined tabular structures known as tables, which consist of rows and columns
        - provide a powerful query language, flexible indexing capabilities, strong integrity controls, and the ability to combine data from multiple tables in a fast and efficient manner
        - allows vertical scalability by increasing resources and horizontal scalability using Read Replicas for read capacity and sharding or data partitioning for write capacity
        - provides High Availability using Multi-AZ deployment, where data is synchronously replicated

    - **NoSQL Databases (DynamoDB)**

- provides databases that trade some of the query and transaction capabilities of relational databases for a more flexible data model that seamlessly scales horizontally
- perform data partitioning and replication to scale both the reads and writes in a horizontal fashion
- DynamoDB service synchronously replicates data across three facilities in an AWS region to provide fault tolerance in the event of a server failure or Availability Zone disruption

- **Data Warehouse (Redshift)**
  - Specialized type of relational database, optimized for analysis and reporting of large amounts of data
  - Redshift achieves efficient storage and optimum query performance through a combination of massively parallel processing (MPP), columnar data storage, and targeted data compression encoding schemes
  - Redshift MPP architecture enables increasing performance by increasing the number of nodes in the data warehouse cluster

Removing Single Points of Failure

- AWS provides ways to implement redundancy, automate recovery and reduce disruption at every layer of the architecture

- **AWS supports redundancy in the following ways**

  - **Standby Redundancy**
    - When a resource fails, functionality is recovered on a secondary resource using a process called failover.
    - Failover will typically require some time before it completes, and during that period the resource remains unavailable.
    - Secondary resource can either be launched automatically only when needed (to reduce cost), or it can be already running idle (to accelerate failover and minimize disruption).
    - Standby redundancy is often used for stateful components such as relational databases.

  - **Active Redundancy**
    - requests are distributed to multiple redundant compute resources, if one fails, the rest can simply absorb a larger share of the workload.
    - Compared to standby redundancy, it can achieve better utilization and affect a smaller population when there is a failure.

- **AWS supports replication**

- **Synchronous replication**
  - acknowledges a transaction after it has been durably stored in both the primary location and its replicas.
  - protects data integrity from the event of a primary node failure
  - used to scale read capacity for queries that require the most up-to-date data (strong consistency).
  - compromises performance and availability

- **Asynchronous replication**
  - decouples the primary node from its replicas at the expense of introducing replication lag
  - used to horizontally scale the system's read capacity for queries that can tolerate that replication lag.

- **Quorum-based replication**
  - combines synchronous and asynchronous replication to overcome the challenges of large-scale distributed database systems
  - Replication to multiple nodes can be managed by defining a minimum number of nodes that must participate in a successful write operation

- **AWS provide services to reduce or remove single point of failure:**
  - Regions, Availability Zones with multiple data centers
  - ELB or Route 53 to configure health checks and mask failure by routing traffic to healthy endpoints
  - Auto Scaling to automatically replace unhealthy nodes
  - EC2 auto-recovery to recover unhealthy impaired nodes
  - S3, DynamoDB with data redundantly stored across multiple facilities
  - Multi-AZ RDS and Read Replicas
  - ElastiCache Redis engine supports replication with automatic failover

Optimize for Cost

- AWS can help organizations reduce capital expenses and drive savings as a result of the AWS economies of scale

- **AWS provides different options which should be utilized as per use case –**

  - **EC2 instance types** – On Demand, Reserved and Spot
  - **Trusted Advisor or EC2 usage reports** to identify the compute resources and their usage
  - **S3 storage class** – Standard, Reduced Redundancy, and Standard-Infrequent Access
  - **EBS volumes** – Magnetic, General Purpose SSD, Provisioned IOPS SSD

- **Cost Allocation tags** to identify costs based on tags
- **Auto Scaling** to horizontally scale the capacity up or down based on demand
- **Lambda** based architectures to never pay for idle or redundant resources
- **Utilize managed services where scaling is handled by AWS for e.g. ELB, CloudFront, Kinesis, SQS, CloudSearch etc.**

Caching

- Caching improves application performance and increases the cost efficiency of an implementation

    - **Application Data Caching**
        - provides services thats helps store and retrieve information from fast, managed, in-memory caches
        - ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud and supports two open-source in-memory caching engines: Memcached and Redis

    - **Edge Caching**
        - allows content to be served by infrastructure that is closer to viewers, lowering latency and giving high, sustained data transfer rates needed to deliver large popular objects to end users at scale.
        - CloudFront is Content Delivery Network (CDN) consisting of multiple edge locations, that allows copies of static and dynamic content to be cached

Security

- **AWS works on shared security responsibility model:**
-
    - **AWS is responsible for the security of the underlying cloud infrastructure**

    - **you are responsible for securing the workloads you deploy in AWS**

- **AWS also provides ample security features:**
    - IAM to define a granular set of policies and assign them to users, groups, and AWS resources
    - IAM roles to assign short term credentials to resources, which are automatically distributed and rotated
    - Amazon Cognito, for mobile applications, which allows client devices to get controlled access to AWS resources via temporary tokens.
    - VPC to isolate parts of infrastructure through the use of subnets, security groups, and routing controls
    - **WAF to help protect web applications from SQL injection and other vulnerabilities in the application code**
    - CloudWatch logs to collect logs centrally as the servers are temporary
    - CloudTrail for auditing AWS API calls, which delivers a log file to S3 bucket. Logs can then be stored in an immutable manner and automatically processed to either notify or even take action on your behalf, protecting your organization from non-compliance
    - AWS Config, Amazon Inspector, and AWS Trusted Advisor to continually monitor for compliance or vulnerabilities giving a clear overview of which IT resources are in compliance, and which are not

Links for the main topics showed on the document

- Since most of the information showed on this document is taken from Jayendra's blog, http://jayendrapatil.com , I consider it is good to have a section with the links for the main topics mentioned on it from Jayendra Patil blog:
- **Appendix to Jayendra's blog**, http://jayendrapatil.com/
- **AWS Certified Solutions Architect – Associate Feb 2018 Exam Learning Path**, http://jayendrapatil.com/aws-solutions-architect-associate-feb-2018-exam-learning-path/

- **AWS Virtual Private Cloud (VPC),** http://jayendrapatil.com/aws-virtual-private-cloud-vpc/
    - **VPC Endpoints**, http://jayendrapatil.com/aws-vpc-endpoints/
    - **VPC Peering**, http://jayendrapatil.com/aws-vpc-peering/
    - **VPC VPN CloudHub Connections**, http://jayendrapatil.com/aws-vpc-vpn-cloudhub/
    - **VPC NAT**, http://jayendrapatil.com/aws-vpc-nat/
    - **Security Group vs NACLs**, http://jayendrapatil.com/aws-vpc-security-group-vs-nacls/
    - **AWS Bastion Host**, http://jayendrapatil.com/aws-bastion-host

- **AWS Elastic Cloud Compute (EC2)**, http://jayendrapatil.com/aws-ec2-overview/
    - **AWS EC2 Amazon Machine Image,** http://jayendrapatil.com/aws-ec2-amazon-machine-image-ami/
    - **AWS EC2 Instance Types**, http://jayendrapatil.com/aws-ec2-instance-types
    - **AWS EC2 Instance Purchase Options**, http://jayendrapatil.com/aws-ec2-instance-purchasing-option
    - **AWS EC2 Instance Lifecycle**, http://jayendrapatil.com/aws-ec2-instance-lifecycle
    - **AWS EC2 Instance Metadata/Userdata,** http://jayendrapatil.com/aws-ec2-instance-metadata-userdata
    - **AWS EC2 Placement Groups**, http://jayendrapatil.com/aws-ec2-placement-groups

    - **AWS EC2 VM Import/Export**, http://jayendrapatil.com/aws-ec2-vm-importexport
    - **AWS EC2 Network, Enhanced Networking,** http://jayendrapatil.com/aws-ec2-enhanced-networking
    - **AWS EC2 Best Practices,** http://jayendrapatil.com/aws-ec2-best-practices
    - **AWS EC2 Monitoring**, http://jayendrapatil.com/aws-ec2-monitoring
    - **AWS EC2 Troubleshooting,** http://jayendrapatil.com/aws-ec2-troubleshooting-connecting-to-an-instance

- **AWS EC2 Storage**, http://jayendrapatil.com/aws-ec2-storage
    - **AWS EC2 Instance Store Storage,** http://jayendrapatil.com/aws-ec2-instance-store-storage
    - **AWS Elastic Block Store Storage**, http://jayendrapatil.com/aws-ec2-ebs-storage
    - **EBS Volume Types,** http://jayendrapatil.com/aws-ebs-volume-types
    - **EBS Snapshot**, http://jayendrapatil.com/aws-ebs-snapshot
    - **EBS Performance**, http://jayendrapatil.com/aws-ebs-performance

- **AWS Simple Storage Service (S3)**, http://jayendrapatil.com/aws-simple-storage-service-s3-overview
- **Amazon S3 Lifecycle Management**, http://jayendrapatil.com/aws-s3-object-lifecycle-management
- **Amazon S3 Permission**, http://jayendrapatil.com/aws-s3-permisions/
- **Amazon S3 Data Protection**, http://jayendrapatil.com/aws-s3-data-protection/
- **Amazon S3 Best Practices**, http://jayendrapatil.com/aws-s3-best-practices/

- **AWS Cloud Migration Services**, http://jayendrapatil.com/aws-cloud-migration-services/

- **AWS Storage Gateway**, http://jayendrapatil.com/aws-storage-gateway

- **AWS Elastic Load Balancing**, http://jayendrapatil.com/aws-elastic-load-balancing

        - **AWS ELB Monitoring**, http://jayendrapatil.com/aws-elb-monitoring
        - **AWS Application Load Balancer**, http://jayendrapatil.com/aws-elb-application-load-balancer/
        - **AWS Network Load Balancer**, http://jayendrapatil.com/aws-elb-network-load-balancer/
        - **AWS Classic Load Balancer vs Application Load Balancer,** http://jayendrapatil.com/aws-classic-load-balancer-vs-application-load-balancer

- **AutoScaling Group (ASG)**, http://jayendrapatil.com/aws-auto-scaling
-
- **AWS Relation Database Service (AWS RDS),** http://jayendrapatil.com/aws-relational-database-service-rds/
    - **AWS RDS Replication (Multi-AZ and Read Replicas)**, http://jayendrapatil.com/aws-rds-replication-multi-az-read-replica
    - **RDS Storage**, http://jayendrapatil.com/aws-rds-storage/
    - **RDS Security,** http://jayendrapatil.com/aws-rds-security/
    - **RDS Best Practices,** http://jayendrapatil.com/aws-certification-rds-best-practices/

- **AWS DynamoDB,** http://jayendrapatil.com/aws-dynamodb
    - **AWS DynamoDB Secondary Indexes**, http://jayendrapatil.com/aws-dynamodb-secondary-indexes

- **AWS Route 53 Overview,** http://jayendrapatil.com/aws-route-53/
    - **AWS Route 53 Routing Policy,** http://jayendrapatil.com/aws-route-53-routing-policy/

- **AWS CloudFront**, http://jayendrapatil.com/aws-cloudfront/

- **AWS ElastiCache**, http://jayendrapatil.com/aws-elasticache-certification/

- **AWS API Gateway**, http://jayendrapatil.com/aws-api-gateway/

- **AWS Lambda**, http://jayendrapatil.com/aws-lambda/

- **AWS Elastic Beanstalk**, http://jayendrapatil.com/aws-elastic-beanstalk/
- **AWS Redshift**, http://jayendrapatil.com/aws-redshift/
- **AWS Kinesis,** http://jayendrapatil.com/aws-kinesis/
- **AWS Simple Email Service (SES)**, http://jayendrapatil.com/aws-simple-email-service-ses/
- **AWS Simple Notification Service (SNS)**, http://jayendrapatil.com/aws-sns-simple-notification-service/
- **Amazon Simple Queue Service (SQS)**, http://jayendrapatil.com/aws-sqs-simple-queue-service/
- **AWS SWF – Simple WorkFlow service**, http://jayendrapatil.com/aws-swf/
- **Amazon Elastic MapReduce (EMR),** http://jayendrapatil.com/aws-emr-certification
- **AWS EC2 Container Service ECS,** http://jayendrapatil.com/aws-ec2-container-service-ecs
- **AWS Directory Services**, http://jayendrapatil.com/aws-directory-services/
- **AWS CloudFormation**, http://jayendrapatil.com/aws-cloudformation/
- **AWS Config,** http://jayendrapatil.com/aws-config/
- **AWS CloudWatch,** http://jayendrapatil.com/aws-cloudwatch-overview/
- **CloudWatch Monitoring Supported AWS Services**, http://jayendrapatil.com/cloudwatch-monitoring-supported-aws-services/

- **Identity & Access Management (IAM)**, http://jayendrapatil.com/aws-iam-overview/
    - **AWS IAM Role**, http://jayendrapatil.com/aws-iam-role/
    - **IAM Role – Identity Providers and Federation**, http://jayendrapatil.com/iam-role-identity-providers-federation/
    - **IAM Policy and Permissions,** http://jayendrapatil.com/aws-iam-access-management/
    - **AWS IAM Roles vs Resource Based Policies,** http://jayendrapatil.com/aws-iam-roles-vs-resource-based-policies/
    - **AWS IAM Best Practices,** http://jayendrapatil.com/aws-iam-best-practices/

- **AWS CloudTrail**, http://jayendrapatil.com/aws-cloudtrail/

- **AWS Key Management Service – KMS**, http://jayendrapatil.com/aws-key-management-service-kms/

- **AWS CloudHSM**, http://jayendrapatil.com/aws-cloudhsm/

- **AWS Trusted Advisor**, http://jayendrapatil.com/aws-trusted-advisor-categories/

- **AWS OpsWorks,** http://jayendrapatil.com/aws-opsworks/

- **AWS WAF (Web Application Firewall)**, http://jayendrapatil.com/aws-waf/

- **Whitepapers**
    - **Architecting for the Cloud - Best Practices,** http://jayendrapatil.com/aws-architecting-for-the-cloud-best-practices-whitepaper
    - **AWS Security,** http://jayendrapatil.com/aws-security-whitepaper-overview/
    - **AWS DDoS Resiliency – Best Practices,** http://jayendrapatil.com/aws-ddos-resiliency-best-practices-whitepaper-overview/

      AWS Disaster Recovery, http://jayendrapatil.com/aws-disaster-recovery-whitepaper/

Write your notes here :)