→ Cloud computing provides a simple way
- to access servers, storage, database and a broad set of application services over the internet.

→ AWS : Amazon Web Services (AWS)
→ owns and maintains the n/w connected hardware required for these application services, while you provision and use what you need via a web application.

AWS platform and AWS Management Console.

→ Fundamental services:
- Amazon Elastic Compute Cloud (EC2)
- Amazon Virtual Private Cloud (VPC) } Module 2.
- Amazon Simple Storage Services (S3)
- Amazon Elastic Block Store (EBS)

→ Security measures - AWS Identity and Access Management (IAM) } Module 3
- AWS database - Amazon DynamoDB, Amazon Relational Database Service (RDS) } Module 4
- AWS Elasticity & Management Tools: } Module 5.
  - Auto Scaling
  - Amazon Cloudwatch
  - Elastic load Balancing (ELB)
  - AWS Trusted Advisor

: AWS Customers:
- Enterprise Customers
- Startup Customers
- Public sector Customers

: AWS Management Console: (Need an Account to explore).
: userID
: account
  ↳ billing dashboard
  ↳ Security credentials
  ↳ Role
: Region
  → resources pertaining to this region selected
: AWS services → search by service names eg: EC2, Route,
  → categorized based on group under services

Advantages & Benefits of AWS Cloud Computing

1. Trade Capital Expense for Variable expense.
2. Benefit from massive economies of scale.
3. Stop guessing capacity (Auto scaling) -> on demand. automated
4. Increase speed and agility. (fail early and fail fast)
5. Stop spending money on running and maintaing data centers
6. Go global in minutes.

Traditional Infrastructure		Amazon Web Service			
Firewalls ACLs Administrators	<Security>	Security Groups	NACLs	AWS IAM.	
Routers N/w pipeline Switch	<Networking>	ELB	VPC		
On-Premises Servers	<Servers>	AMI (image) → Amazon EC2 Instances			
DAS SAN NAS RDBMS	<Storage & Database>	Amazon EBS	Amazon EFS	Amazon S3	Amazon RDS

AWS Cloud Computing:

Applications → Virtual Desktops | Collaboration and Sharing

Platform Services

Database	Analytics	App Services	Development & Management	Mobile Services
Relational	Cluster Computing	Queueing	Container	Identity
No SQL	Real time	Orchestration	Dev/ops Tools	Sync
Caching	Data Warehouse	App Streaming	Resource Templates	Mobile Analytics
	Data Workflows	Transcoding	Usage Tracking	Notification
		Emoji	Monitoring & Logs	
		Search		

Foundation Services

- **Compute** (Virtual, autoscaling and load balancing) — Amazon EC2, AWS Lambda, Amazon ECS Container Service, Elastic Load Balancing, AWS Elastic Beanstalk, Auto Scaling
- **Network** — Amazon VPC, Amazon Route 53, AWS Direct Connect
- **Storage** (Object, Block & Archive) — Amazon S3, Amazon CloudFront, Amazon Glacier, Amazon Elastic File System, AWS Storage Gateway, AWS Import/Export Snowball
- **Security & Identity** — AWS Identity & Access Management, AWS KMS, AWS Directory Service, AWS Cloud HSM, AWS WAF
- **Application** — Amazon WorkDocs, Amazon WorkSpace, Amazon WorkMail

Infrastructure — Regions | Availability Zone | Edge locations

AWS Global Infrastructure :

Region Decision : latency, Regulatory & Compliance, Cost, Availability

atleast 2 Availability Zones (AZs) + Edge Locations
(Amazon Route 53, Amazon CloudFront)
→ reduce latency
→ improve performance

clusters of data centers (isolated) for tolerance

AWS Foundational Services:

★ AWS Core Services:

1. Amazon Elastic Compute Cloud (EC2) :

→ instance which is a virtualized servers that runs in AWS datacentre
→ resizable compute capacity
→ complete control of your computing resources
→ Reduces the time required to obtain and boot new server instances

→ Elastic → Scale capacity as computing requirements change
: Low Cost → Pay only for capacity that is actually used
: Linux/Windows Choose Linux/Windows
: AWS Global infrastructure → Deploy across AWS regions & availability zones for reliability

Steps involved / How to launch an Amazon EC2 instance via Web Console:

1. Determine the AWS Region
2. Launch an Amazon EC2 instance from pre-configured Amazon Machine Image (AMI)
3. Choose an instance type based on CPU, memory, storage & n/w requirements
4. Configure n/w, IP address, security groups, storage volume, tags, & key pair.

2. Amazon Machine Image (AMI) :

→ template for the root volume for the instance (eg: OS, application server)
→ Launch permissions that control which AWS accounts can use the AMI to launch instances
→ A block device mapping that specifies the volumes to attach to the instance when it's launched.

Select based on: → Region
→ OS
→ Architecture
→ Launch permission
→ Storage for the root Device

→ Can launch multiple instances on a single AMI.
→ in Amazon easy to public cloud, & Amazon Virtual private Cloud (VPC) in availability zone within a region.
→ customers can then deploy to multiple availability zones within a region.
↳ Snapshots & Buckets can be used.

Amazon EC2 Instance Store

⇒ Data Stored on a local instance store persists only as long as the instance is alive

→ Storage is ephemeral

→ keep data, underlying host changes, persists only during the life of the instance

- Cannot be stopped.

Amazon EBS

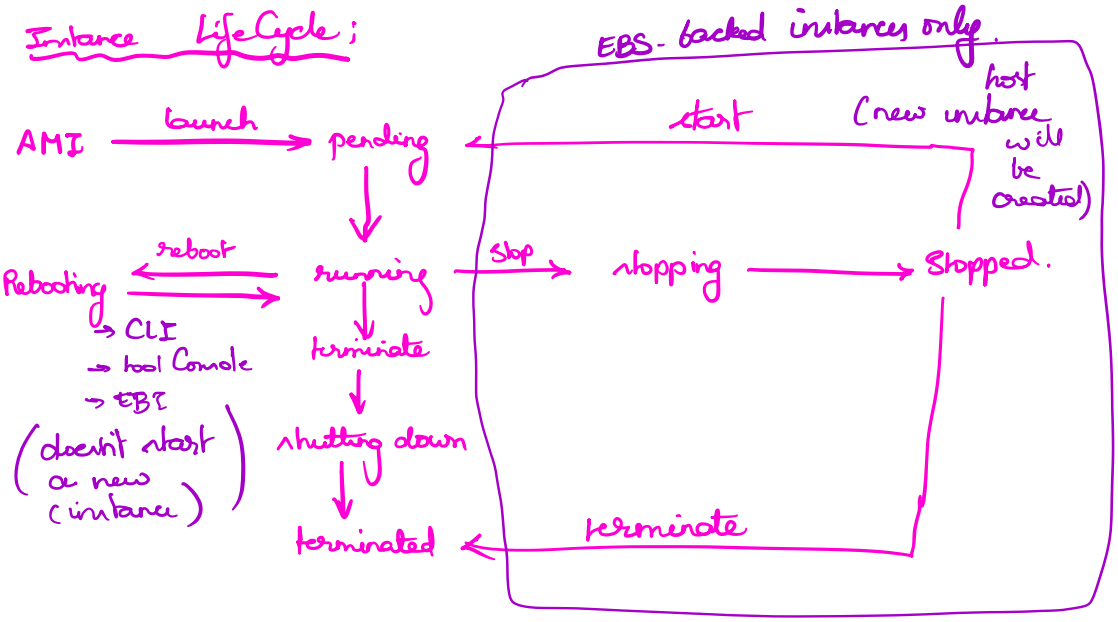
→ Data Stored on an Amazon EBS volume can persist independently of the life of the instance

→ Storage is persistent.

→ The root volume is deleted when the instance terminates. Data on any other Amazon EBS volumes persists after instance termination

- Can be stopped.

Instance LifeCycle;



1. General Propose → low-traffic webiks & web applications, small dbs / mid-size dbs.
2. Compute optimized → high performance front-end fleets, video encoding
3. Memory optimized → high performance dbs, distributed memory caches
4. Storage optimized → data warehousing, log / data processing applications
5. GPU instances → 3D application streaming, machine learning

Instance Metadata

1. data about your instance
2. Can be used to configure or manage a running instance
3. Not much security.

curl http://169.254.169.254/latest/user-data

GET http://169.254.169.254/latest/user-data

or

```
#!/bin/sh
yum -y install httpd
chkconfig httpd on
/etc/init.d/httpd start.
```

#!/bin/sh → shell script.

yum -y install httpd → install apache web server

chkconfig httpd on → enable web server

/etc/init.d/httpd start. → start web server

Instance User data

- Can be passed to the instance at launch
- Can be used to perform common automated configuration tasks
- Runs scripts after the instance starts

cloud-init , EC2Config service.

3. Storage Services: Amazon S3 and Amazon EBS:

1. Amazon Simple Storage Service (Amazon S3):

- storage for the internet
- naturely online, HTTP access
- Store and retrieve any amount of data, any time from anywhere on web
- highly scalable, reliable, fast and durable
  - unlimited number of objects in a bucket
  - 5 TB objects, no bucket size limit
  - 99.99% durability & 99.99% availability of objects over a given year
  - HTTP/s endpoint to store and retrieve any amount of data
  - highly scalable/reliable/fast/inexpensive
  - optional server-side encryption using AWS or customer-managed provided client-side encryption
  - access log for audit
  - standards-based REST & Soap interfaces
- Amazon S3 store data as Objects within buckets.
  - Object → composed of a file and optionally any metadata that describes that file
  - 100 buckets in each account
  - Can control access to the bucket & its objects.
  - Bucket → amazon S3 namespace, identify the account, scale, unit of aggregation
    - globally unique bucket names, regardless of the AWS region in which they were created.

Object Key → uniquely identifier of an object in a bucket:

eg: http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.html

bucket — Object key

→ Amazon S3 Security: (eg: SoundCloud → S3 ← Glacier) (standard)

→ can control access to buckets and objects with:
1. Access Control list (ACLs) → (account level)
2. Bucket policies → (access)
3. Identity and Access Management (IAM) policies → (within account)

→ can upload and download to Amazon S3 via SSL encrypted endpoints
→ can encrypt data using AWS SDKs.
→ (Glacier → backups) → long term, infrequent data., infrequent access (IA). standard.

→ Amazon S3 versioning:
→ 3 states of version → unversion, version

2. Amazon Elastic Block Store (EBS)

→ persistent block level storage volume
→ offer consistent & low-latency performance
→ automatically replicated within its availability zone
→ snapshot stored durably in Amazon S3.

EBS lifecycle:

Unit amount of unused space → (createVolume) → create → (AttachVolume) → attach → attached & in use → format from Amazon EC2 instance OS. mount formatted drive → create Snapshot → Detach → (DeleteVolume) Delete / DetachVolume

→ EBS magnetic volumes from 1 GB to 1 TB
→ EBS General Purpose (SSD) & Provisioned IOPS (SSD) → 16 TB
→ encrypted EBS volumes
→ create point-in-time snapshots of EBS volumes
→ frequent, durable persistence
→ highly reliable/available
→ disk drive

3. Amazon EC2 Instance Storage:

→ local, complimentary direct attached block storage resource
→ no persistence
→ SSD / magnetic
→ all data is automatically deleted when an EC2 instance stops, fails & is terminated.

Characteristics :	Reboo	Stop/Start (EBS backed instance only)	Terminate
1. Host Computer	instance stays on the same host	runs on new host	NA
2. Private & public IP address	stays the same	instance keeps the private IP address & gets a new public IP address	NA
3. Elastic IP address (EIP)	EIP remains associated with the instance	EIP remain associated with the instance	EIP is dissociated from the instance
4. Instance store volumes.	data is preserved	The data is erased	data is erased
5. EBS volume	volume is preserved	volume is preserved	volume is deleted by default
6. Billing.	instance billing hr doesn't change	stop incurring charges as soon as state is changed to stop	stop incurring charges as soon as state is changed to shutting down.

4. **Networking Amazon VPC**
Amazon Virtual Private Cloud (VPC)

→ provision a private, isolated virtual n/w on AWS Cloud
→ have complete control over your virtual networking environment.

: Subnet :→ range of IP addresses in your VPC. must reside within one AZ & cannot span zone.
launch resources into a subnet that you select.

↳ public subnet :→ used for resources that will be accessed over the internet.
↳ private subnet → used for resources that won't be accessible over the internet. (Backend).

→ logical isolation, VPC customization. , Network ACLs act as a firewall for associated subnets controlling both inbound & outbound traffic at the subnet level.

: VPN Connections:
- AWS hardware VPN
- AWS Direct Connect
- AWS VPN CloudHub
- Software VPN.

BUILD YOUR OWN INFRASTRUCTURE:

Console → VPC Service

(Start VPC Wizard)

→ with public & private subnets.

: CIDR block → default

VPC name → demo (change) → find by name

change: public subnet IPv4 CIDR

→ eg: 10.0.1.0/24.

change: private subnet IPv4 CIDR

→ eg: 10.0.3.0/24

AZ → No Preference → us-east-1a (for both)

change: details of your NAT instance; (or can use NAT gateway)

instance type – m1.small

Key pair → (already created for us, select).

→ VPC subnets, NAT & other configurations are made.

→ Subnets → Create subnet. (public & private)

Name tag: Public Subnet 2 / Private Subnet 2.

VPC → can give our VPC

same availability zone as VPC Service.

→ CIDR block → 10.0.2.0/24. / 10.0.4.0/24.

VPC.

zone A

VPC zone B.

→ Route Table → tag name for main = Yes (public & private)

Routes associated with this.

→ Elastic interface, VPC created EC2 instance.

→ Subnet Association

→ select the ones you need. (private)

→ Security Group → Under Security (default: no inbound rules)

→ name, Web Security Group.

desc: Enable http access

enable http traffic (type: HTTP)

port: 80

any source → Protocol: TCP

range: 0.0.0.0/0

→ To start a service on VPC

→ EC2 Service

→ brand new instance

1] AMI select the image of OS / Linux, red hat, Suse linux etc.;)

2] Instance Type → default configurations

3] → Configure Instance Details

: number of instance = 1

n/w → created VPC service

subnet (created subnets)

auto-assign Public IP → enable

IAM role → optional

→ advanced details.

→ bootstrapping:

→ user details like apache web server

Eg: /etc/init.d/httpd start

```
if [ ! -f /var/www/html/lab2-app-tar.gz ]; then
cd /var/www/html
wget https://s3-us-west-2-aws-training.s3.a...
ESS
tar xvfz
chown apache:root /var/www/html/sdi.config.php
fi.
```

4) Add Tags → storage → (8 GiB) default.

(tags → naming to storage) eg: Web server

5) Configure Security Group;

→ select an existing security group.

→ review & launch : → { 22 → ssh for remote).

→ create a new private key (.ppk)

test with public DNS name; that was given (url of web server).

o

Shared Security Model

Owners

Customer Data.

Platform, Application, Identity, & Access Management.

OS, N/w & Firewall Configuration

Customers

Client-side Data Integrity Authentication

Server-side Encryption (File system and/or Data)

N/w Traffic Protection (Encryption/Integrity/Identity)

Write the Rules

AWS

Foundation Services:

Compute Storage Database Network.

AWS Global Infrastructure

Availability Zone. Regions

Edge Locations.

Enforce the Rules

→ SSL/TLS Endpoints:
- → secure transmission
- → instance Firewalls.
- → N/w control → N/w access control range for traffic flows.

→ AWS Multi-Tier Security Groups:

AWS Identity and Access Management (IAM):

1. Manage AWS IAM users and their access
2. Manage AWS IAM roles & their permissions
2. Manage federated users and their permissions

→ SDK, Tool, Console (IAM Authentication)

Java, Python, .NET (SDK, Tool,)

→ Create an IAM user
- ⇒ AWS Access Key ID: *
- AWS Secret Access Key *
- Default region name
- Default output format.

→ IAM user Management → Groups: (AWS Account)

Dev TestDev

- Authorization (policy-JSON) → IAM Roles → no password is needed

User Role Group

policy (no credentials)

→ AWS Resources (to get different access)
  ↳ roles assumed.
  ↳ users assumed
  - assign only when creating and instance.

→ Temporary Security Credentials (AWS STS).
  ↳ Access Key Id,
  ↳ Security Access Key
  ↳ Session Token
  ↳ Expiration

AWS IAM Best Practices:

→ Delete AWS account (root) access keys
→ Create individual IAM users.
→ Use groups to assign permission to IAM users
→ Grant least privilege
→ Configure a strong password policy
→ Enable MFA for privilege users
→ Use roles for application that run on Amazon EC2 instances
→ Delegate by using roles instead of by sharing credentials
→ Rotate credentials regularly
→ Remove unnecessary users and credentials
→ Use policy conditions for extra security
→ Monitor activity in your AWS account.

AWS Database

- Amazon DynamoDB
- Amazon ElasticCache
- Amazon RDS
- Amazon Redshift
- AWS Database Migration Service

→ Data Storage Consideration:
- data format
- data size
- Query frequency
- Data access speed
- Data retention period.

	SQL	NoSQL
Data Storage	Rows & Columns	Key-value
Schemas	Fixed	Dynamic
Querying	using SQL	focused on collection of documents
Scalability	Vertical	Horizontal.

→ Amazon Relational Database Service (RDS)

- Cost-efficient
- resizable Capacity
- manages admin tasks
- access to full capabilities
- simple & fast to deploy
- fast to scale
- Compatible with your application

→ monitor
- multi AZ
- automatic backup
- increase the I/O capacity
- set a TTL < 30s
- Test failover

→ Amazon Aurora, MySQL, MariaDB, Microsoft SQL Server, Oracle, PostgreSQL

Building Blocks:
→ DB instances
→ isolated db env in the cloud
- contain multiple user-created dbs
- retention period up to 35 days
- allows Manual Snapshots (initiated by the user)
→ persist until the user deletes them
→ portable

→ Amazon RDS apis
→ Tools (Command line)
→ Console

- Run DB instance in an Amazon VPC, grant access by IAM policies
- SSL credentials
- auto failover (standby instance for maintainance, deploy time)
- DB parameter & Option Groups

Amazon DynamoDB:
- no limits
- using SSDs (fast)
- easy to provision & change (request capacity)
- AWS fully managed service

→ Table → collection of Items
→ attributes (name-value pairs) (single, json etc)

Primary key for Table (unique)
↳ Partition key (as input to hash function, op of hash function determines where to store)
↳ Sort key

- local secondary key (non key value)

↑ all these partitions are sorted using sort key

Build Database Server;

Console → VPC → Security Groups. (DB Security Groups)
(modify inbound for ports)
(My SQL Aurora)
TCP, 3306 port,
source : sg-e187309e (sql

Source – RDS
↳ subnet groups.

1) Create DB Subnet group; (add three private subnets created in new)

2) Launch Select Engine → MySQL → select.
↳ Production → MySQL → next.
: DB details (instance class change).
Multi AZ deployment = Yes

3) DB Instance Identifier (unique name)
master username, passwords

4) Change Configure Advanced Setting :
VPC, AZ, Subnet group
Database Name.
Monitoring Yes.
(endpoint → will be generated) (use that in EC2 instance where we have RDS access (no port)
database → DB name name as used.
username → master.
password →

Elasticity & Management Tools:

- auto Scaling: → right no of instances to run.
  - fault Tolerance, availability, cost management.
  - launch Configure. (AMI ID, instance type, key pair, security groups, block device mapping, user data.
  - auto scaling group (logical)
    - minimum, desired capacity, maximum size.
- elastic load balancing
  - distribute
  - health checks
  - routing & load balancing.
  - API/CLI/Console to manage.
  - load balancer Types:
    - Internet facing
    - Internal
    - https load balancer (SSL header)
- cloud Watch
  - monitoring service
  - centralized
  - visibility across
    - resource utilization
    - operational performance
  - custom application specific
  - Console/API/SDK/CLI
  - set alarms, notify & view graphs
  - metric repository

Artifacts:

/Users/nanditha/Desktop/CanadaDocs/Hobbies/ReadMe/AWS

Fundamentals		
Learning Resource	**Duration**	**Type**
AWS Technical Essentials	4 hour 30 minutes	Digital Training

Course structure:
https://explore.skillbuilder.aws/learn/course/1851/play/45289/aws-technical-essentials-104;lp=1044

NOTES:

Overview: Companies like AWS own and maintain data centers and provide virtual data center technologies and services to companies and users over the internet.

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

Six advantages of cloud computing :

1. you **pay only when you use** computing resources, and pay only for how much you use.
2. AWS can achieve **higher economies of scale**, which translates into lower **pay as-you-go** prices.
3. **Eliminate guessing on your infrastructure capacity needs.** You can access as much or as little capacity as you need, and scale up and down as required with only a few minutes notice.
4. you **reduce the time to make resources available** to your developers from weeks to minutes. This results in a dramatic **increase in agility for the organization,** since the **cost and time it takes to experiment and develop is significantly lower.**
5. Cloud computing lets you focus on your customers, rather than on the heavy lifting of racking, stacking, and powering physical infrastructure. This is often referred to as **undifferentiated heavy lifting**.
6. Applications can be deployed in multiple Regions around the world with a few clicks. i.e you can provide **lower latency and a better experience for your customers at a minimal cost**.

Types of cloud computing :

1. **Infrastructure as a Service (IaaS)** - It typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space. IaaS gives you the highest level of flexibility and management control over your IT resources.
2. **Platform as a Service (PaaS) -** PaaS removes the need for you to manage underlying infrastructure (usually hardware and operating systems), and allows you to focus on the deployment and management of your applications. Don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.
3. **Software as a Service (SaaS) -** SaaS provides you with a complete product that is run and managed by the service provider. Don't have to think about how the service is maintained or how the underlying infrastructure is managed. You only need to think about how you will use that particular software.

Cloud computing deployment models:

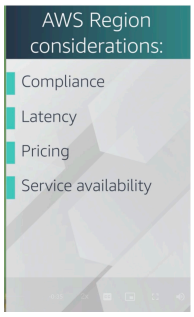
1. **Cloud -** A cloud-based application is fully deployed in the cloud and all parts of the application run in the cloud.
2. **Hybrid** - A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure to extend, and grow, an organization's infrastructure into the cloud while connecting cloud resources to the internal system.
3. **On-premises -** The deployment of resources on-premises, using virtualization and resource management tools, is sometimes called the "private cloud." On-premises deployment doesn't provide many of the benefits of cloud computing but is sometimes sought for its ability to provide dedicated resources. In most cases this deployment model is the same as legacy IT infrastructure while using application management and virtualization technologies to try and increase resource utilization

Working Demo Example: Corporate directory application

- Use AWS services to architect a scalable, highly available, and cost-effective infrastructure to host the corporate directory application.

Step 1: AWS Global Cloud Infrastructure

In AWS, this physical infrastructure makes up the AWS Global Infrastructure, in the form of Regions. Regions are clusters of Availability Zones. Availability Zones are clusters of data centers.



AWS Regions and Availability Zones:

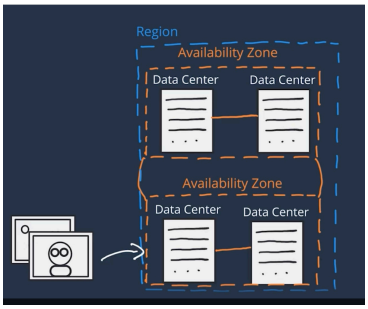
Availability Zones. **ex**: us-east-1a: An AZ in us-east-1 (N. Virginia Region)

•

Regions are geographic locations worldwide where AWS hosts its data centers.

- Each AWS Region is associated with a geographical name and a Region code.
- Data is not replicated from one Region to another, without explicit customer consent and authorization.
- Inside every Region is a cluster of **Availability Zones (AZs).**

Availability Zone:



- An AZ consists of one or more data centers with redundant power, networking, and connectivity.
- These data centers operate in discrete facilities in undisclosed locations. They are connected using redundant high-speed and low-latency links.
- **Maintain resiliency** : At a minimum, you should use two AZs. That way, if an AZ fails, your application will have infrastructure up and running in a second AZ to take over the traffic.

AWS Service Endpoints:

- To connect programmatically to an AWS service, you use an **endpoint.**
- An endpoint is the URL of the entry point for an AWS web service. Most Amazon Web Services offer a Regional endpoint that you can use to make your requests.
- Some AWS services offer endpoints that support Federal Information Processing Standard (**FIPS**) 140-2 in some Regions. With FIPS endpoints, the minimum requirement is **TLS 1.2.** AWS revoked the ability to use TLS 1.0 and TLS 1.1 on all FIPS endpoints in all Regions as of March 31, 2021.
- Some AWS services offer dual stack endpoints, so that you can access them using either IPv4 or IPv6 requests.

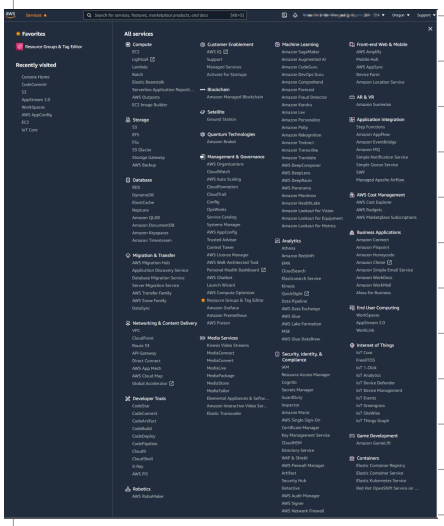
protocol://service-code.region-code.amazonaws.com

For example, https://dynamodb.us-west-2.amazonaws.com is the endpoint for the Amazon DynamoDB

service in the US West (Oregon) Region.

NOTES:

1. 31 Launched Regions each with multiple Availability Zones (AZs)
2. 99 Availability Zones
3. 410+ Points of Presence 400+ Edge Locations and 13 Regional Edge Caches
4. 32 Local Zones , 29 Wavelength Zones for ultralow latency applications
5. 245 Countries and Territories Served
6. 115 Direct Connect Locations
7. The AWS SDKs and the AWS Command Line Interface (AWS CLI) automatically use the default endpoint for each service in an AWS Region.



Step 2: Interaction with Virtual Infrastructure using APIs

A)web application

B)aws-shell is a command-line shell program

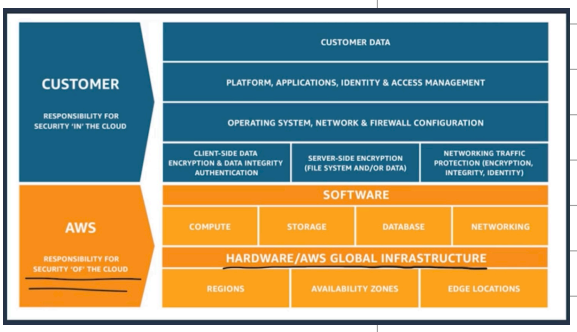
```
aws ec2 describe-instances
```

you will get the following response:

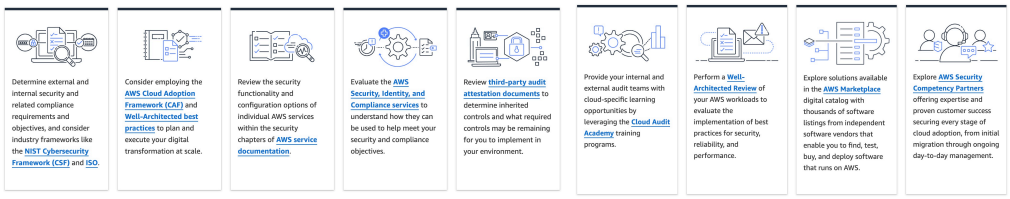
```
{
"Reservations": [
{
"Groups": [],
"Instances": [
{

"AmiLaunchIndex": 0,
```

C)Easily develop applications on AWS in the programming language of your choice

```
import boto3
ec2 = boto3.client('ec2')
response = ec2.describe_instances()
print(response)
```



Step 3: Security and the AWS Shared Responsibility Model

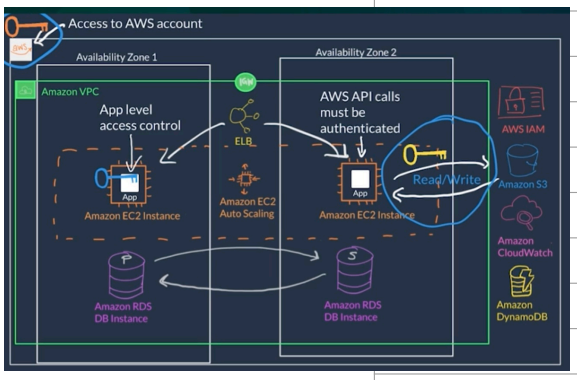


The following exercises can help customers in determining the distribution of responsibility based on specific use case:

Device	Description	Supported Devices
Virtual MFA	A software app that runs on a phone or other device that provides a one-time passcode. These applications can run on unsecured mobile devices, and because of that, they might not provide the same level of security as hardware or U2F devices.	Authy, Duo Mobile, LastPass Authenticator, Microsoft Authenticator, Google Authenticator
Hardware	A hardware device, generally a key fob or display card device, that generates a one-time, six-digit numeric code.	Key fob, display card
U2F	A hardware device that you plug in to a USB port on your computer.	YubiKey

Step4: Protect the AWS Root Use

1. You can have up to eight MFA devices of any combination of the currently supported MFA types assigned to a user at a time with your AWS account root user and IAM users.
2. **A hardware TOTP token** generates a six-digit numeric code based upon a time-based one-time password (TOTP) algorithm.
3. **FIDO security keys** are a type of multi-factor authentication (MFA) device that you can use to protect your AWS resources.FIDO2 is an open authentication standard and an extension of FIDO U2F, offering the same high level of security based on public key cryptography. FIDO2 consists of the W3C Web Authentication specification (WebAuthn API) and the FIDO Alliance Client-to-Authenticator Protocol (CTAP), an application layer protocol.
4. **Virtual authenticator apps:** You can use a **phone or other device as a virtual multi-factor authentication (MFA) device**. App that is compliant with RFC 6238, a standards-based TOTP (time-based one-time password) algorithm



5. The root user can perform all actions on all resources inside an AWS account by default. This is in contrast to creating new IAM users, new groups, or new roles

Step 5: AWS Identity and Access Management (IAM) -AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.

- Manage access to your AWS account and resources. It also provides a centralized view of who and what are allowed inside your AWS account (authentication), and who and what have permissions to use and work with your AWS resources (authorization).
- An **IAM user** represents a person or service that interacts with AWS.
- **IAM user credentials -** can provide them with the following types of access:
  - Access to the AWS Management Console
  - Programmatic access to the AWS Command Line Interface (AWS CLI) and AWS application programming interface (AWS API).
- recommend you only use IAM users for use cases not supported by federated users.
- An **IAM group** is a collection of users. Makes it possible to give permissions to multiple users at once.features of groups:
  - Groups can have many users.
  - Users can belong to many groups.
  - Groups cannot belong to groups.
- **IAM policies** - The way you grant permissions in IAM is by using IAM policies i.e To manage access and provide permissions to AWS services and resources, you create IAM policies and attach them to IAM users, groups, and roles.

Element	Description	Required	Example
`Effect`	Specifies whether the statement results in an allow or an explicit deny	√	`"Effect": "Deny"`
`Action`	Describes the specific actions that will be allowed or denied	√	`"Action": "iam:CreateUser"`
`Resource`	Specifies the object or objects that the statement covers	√	`"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"`

Example:

```
 {
"Version": "2012-10-17",
"Statement": [{
"Effect": "Allow",
```

```
"Action": [
"iam: ChangePassword",
"iam: GetUser"
]
"Resource": "arn:aws:iam::123456789012:user/${aws:username}"
}]

}
```

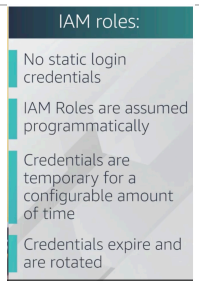
- **IAM Identity Center** provides:
  - A central set of identities and assignments
  - Access to accounts across an entire AWS Organization
  - Connection to your existing identity provider
  - Temporary credentials
  - Multi-factor authentication (MFA)
  - Self-service MFA configuration for end-users
  - Administrative enforcement of MFA usage
  - Single sign-on to all AWS account entitlements

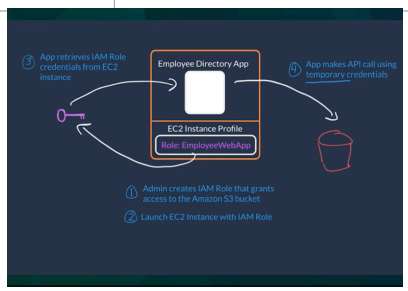
When to create an IAM user (instead of a role): USE CASES:

1. **Workloads that cannot use IAM roles :** In some situations, you can't use IAM roles to provide temporary credentials, such as for WordPress plugins. In these situations, use IAM user long-term access keys for that workload to authenticate to AWS.
2. **Third-party AWS clients :** [tools that don't support access with IAM Identity Center, such as third-party AWS clients or vendors that are not hosted on AWS, use IAM user long-term access keys.]
3. **AWS CodeCommit access:** If you are using CodeCommit to store your code, you can use an IAM user with either SSH keys or service-specific credentials for CodeCommit to authenticate to your repositories. Can configure the git-remote-codecommit utility.
4. **Amazon Keyspaces (for Apache Cassandra) access** : In a situation where you are unable to use users in IAM Identity Center, such as for testing purposes for Cassandra compatibility, you can use an IAM user with service-specific credentials to authenticate with Amazon Keyspaces.
5. **Emergency access** : In a situation where you cannot access your identity provider and you must take action in your AWS account.

When to create an IAM role (instead of a user) : USE CASES:

1. **You're creating an application that runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance and that application makes requests to AWS -** Don't create an IAM user and pass the user's credentials to the application or embed the credentials in the application. Instead, create an IAM role that you attach to the EC2 instance to give temporary security credentials to applications running on the instance.
2. **You're creating an app that runs on a mobile phone and that makes requests to AWS** - Don't create an IAM user and distribute the user's access key with the app. Instead, use an identity provider like Login with Amazon, Amazon Cognito, Facebook, or Google to authenticate users and map the users to an IAM role. The app can use the role to get temporary security credentials that have the permissions specified by the policies attached to the role.
3. **Users in your company are authenticated in your corporate network and want to be able to use AWS without having to sign in again—that is, you want to allow users to federate into AWS -** Don't create IAM users. Configure a federation relationship between your enterprise identity system and AWS.
   A. If your company's identity system is compatible with SAML 2.0, you can establish trust between your company's identity system and AWS.
   B. Create and use a custom proxy server that translates user identities from the enterprise into IAM roles that provide temporary AWS security credentials.



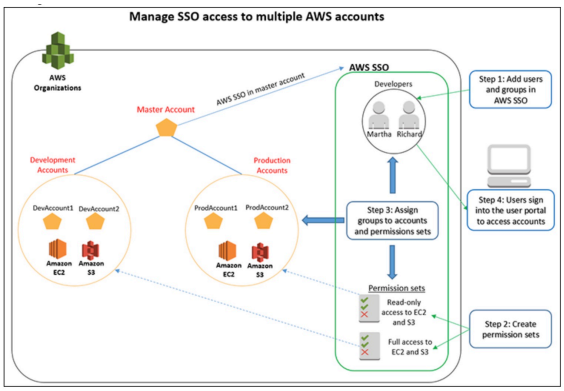


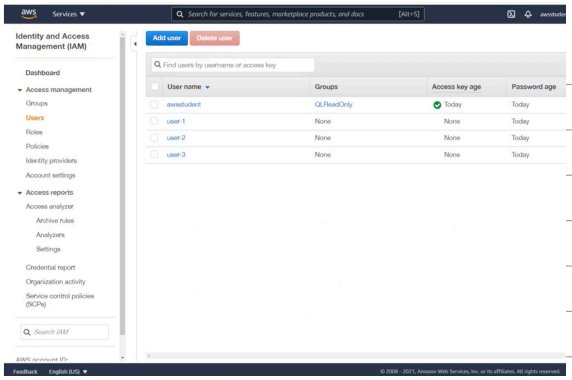
- An **IAM role** is an identity within your AWS account that has specific permissions.

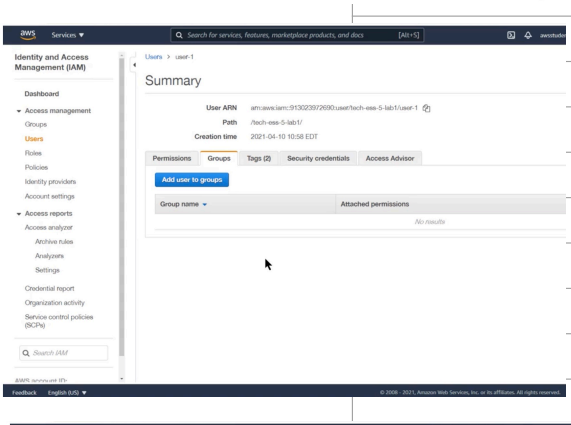
IAM best practices:

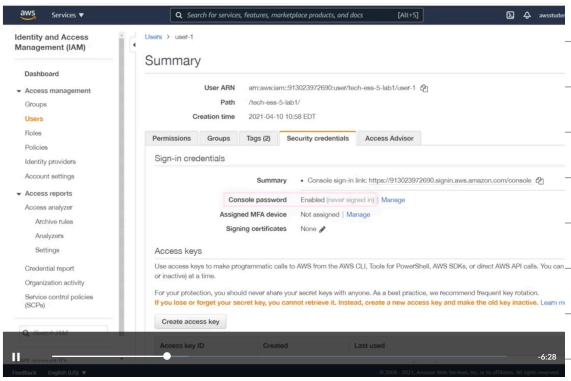
1. **lock down the root user, you can do the following:**
   - Don't share the credentials associated with the root user
   - Consider deleting the root user access keys
   - Enable MFA on the root account
2. **Least privilege** is a standard security principle that advises you to grant only the necessary permissions to do a particular job and nothing more. To implement least privilege for access control,
   - start with the minimum set of permissions in an IAM policy and
   - then grant additional permissions as necessary for a user, group, or role.
3. **IAM is used to secure access to your AWS account and resources.**
   - It provides a way to create and manage users, groups, and roles to access resources in a single AWS account.
   - IAM is not used for website authentication and authorization, such as providing users of a website with sign-in and sign-up functionality.
   - IAM also does not support security controls for protecting operating systems and networks.
4. **Use IAM Roles when possible:**
   - Maintaining roles is more efficient than maintaining users.
   - When you assume a role, IAM dynamically provides temporary credentials that expire after a defined period of time, between 15 minutes and 36 hours.
   - Users, on the other hand, have long-term credentials in the form of user name and password combinations or a set of access keys.
   - User access keys only expire when you or the account admin rotates the keys.
   - User login credentials expire if you applied a password policy to your account that forces users to rotate their passwords.
5. **Consider using an identity provider (IdP)** :
   - Using an IdP, whether it's an AWS service such as AWS Single Sign-On or a third-party identity provider, provides a single source of truth for all identities in your organization.
   - You no longer have to create separate IAM users in AWS. You can instead use IAM roles to provide permissions to identities that are federated from your IdP.
6. **Consider using AWS SSO :** AWS SSO is an IdP that lets your users sign in to a user portal with a single set of credentials. It then provides users access to their assigned accounts and applications in a central location.
   - AWS SSO offers a directory where you can create users, organize them in groups, set permissions across the groups, and grant access to AWS resources.
   - If you're using a third-party IdP, you can sync your users and groups to AWS SSO. This removes the burden of having to re-create users that already exist elsewhere, and it enables you to manage the users from your IdP.
   - AWS SSO separates the duties between your IdP and AWS, ensuring that your cloud access management is not inside or dependent on your IdP.
7. **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions**
8. **Use permissions boundaries to delegate permissions management within an account**
9. **Establish permissions guardrails across multiple accounts**
10. **Require human users to use federation with an identity provider to access AWS using temporary credentials**
11. **Require workloads to use temporary credentials with IAM roles to access AWS**
12. **Require multi-factor authentication (MFA)**
13. **Rotate access keys regularly for use cases that require long-term credential**

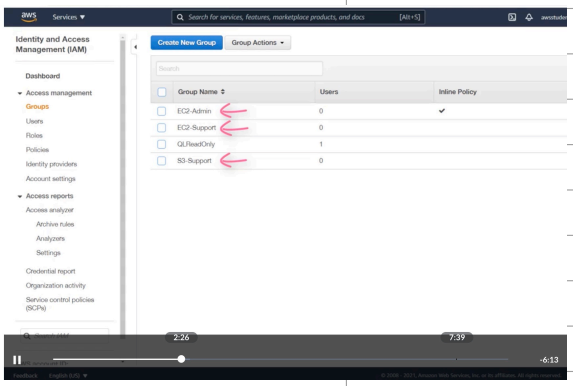
How to Create and Manage Users within AWS Single Sign-On :
https://aws.amazon.com/blogs/security/how-to-create-and-manage-users-within-aws-sso/
https://explore.skillbuilder.aws/learn/course/1851/play/45289/aws-technical-essentials-104;lp=1044

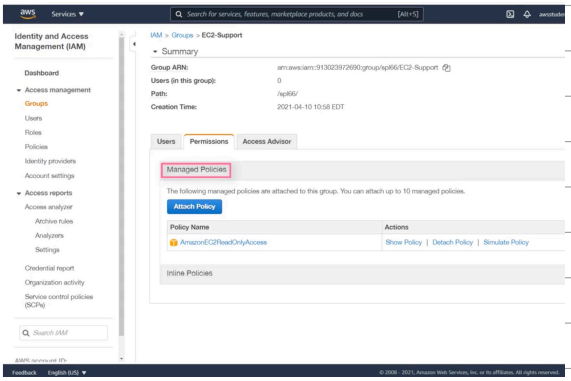


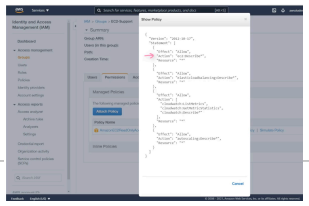


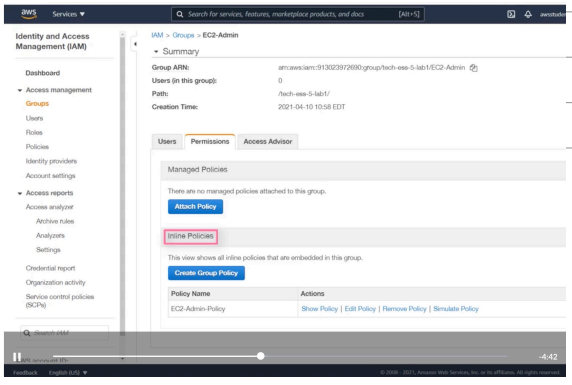


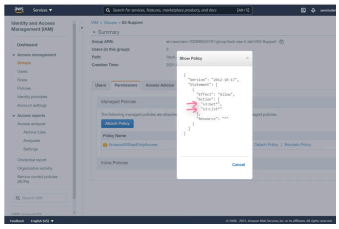




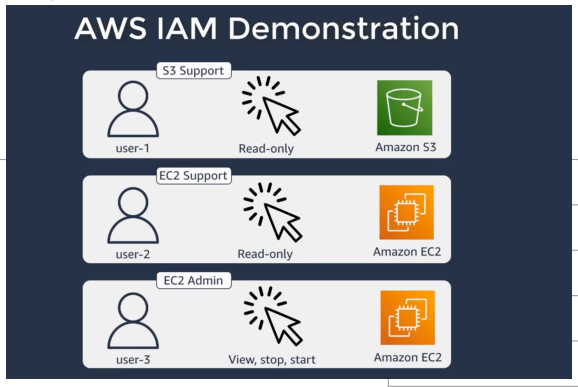


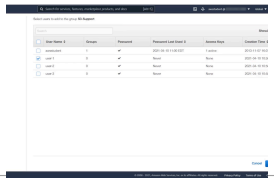


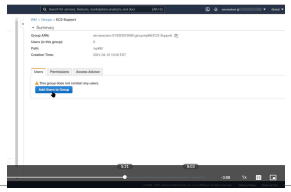


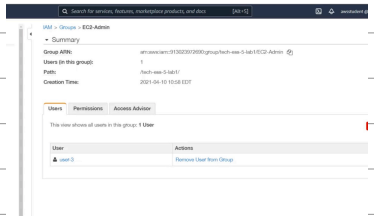


IMplementation:

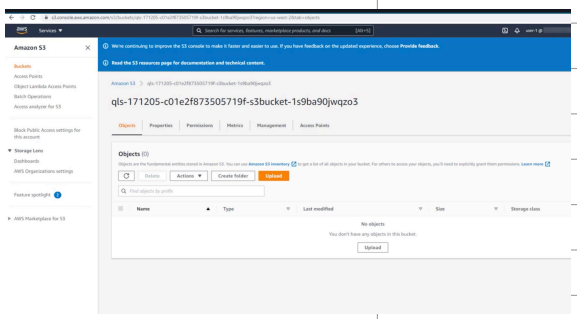


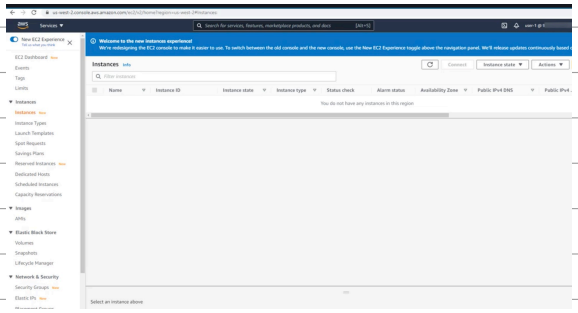


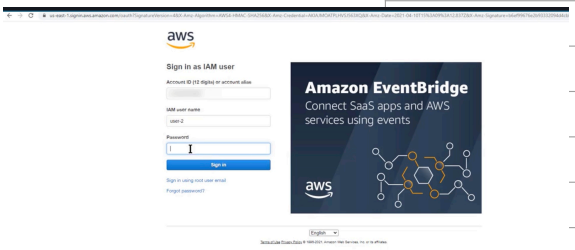


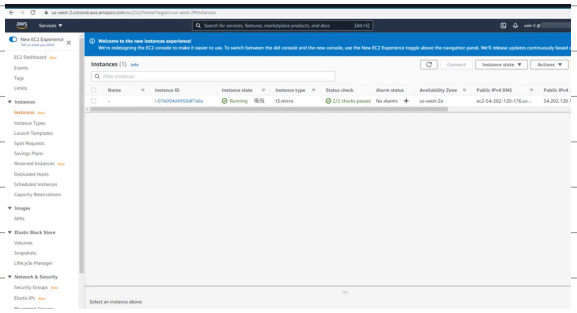


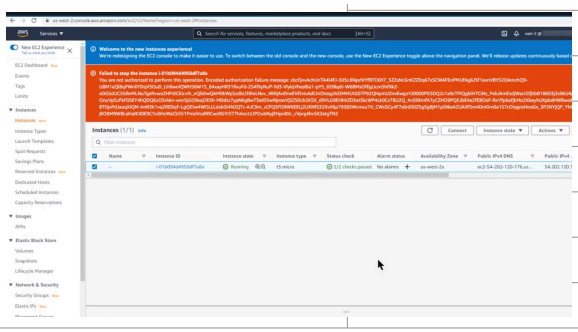
Validation:

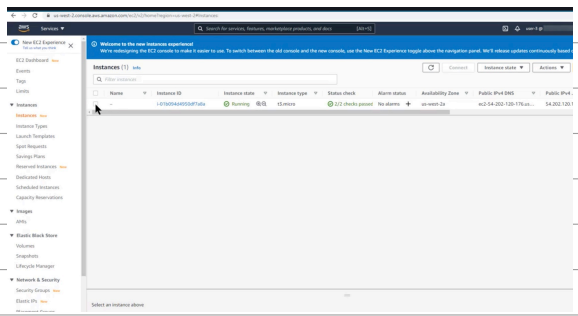


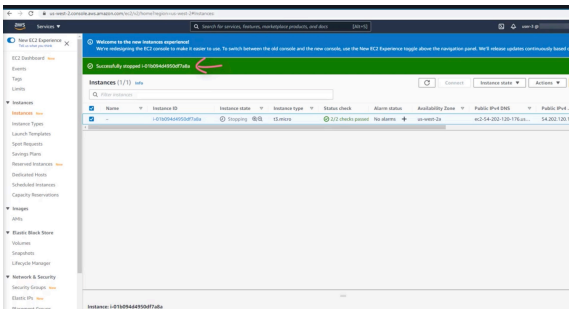




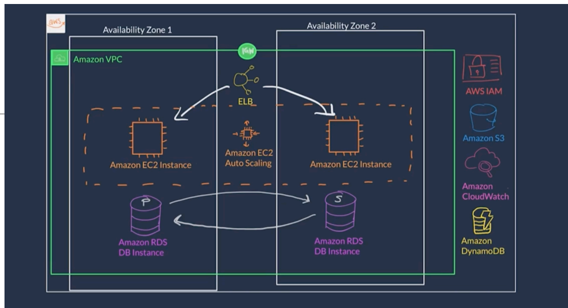


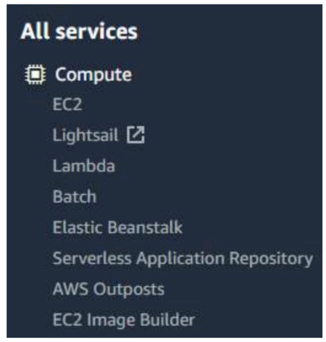






Architecture:





>>
COMPUTE AS A SERVICE
>>

Servers power your application by providing CPU, memory, and networking capacity to process users' requests and transform them into responses. For context, common HTTP servers include:

- ○ Windows options, such as Internet Information Services (IIS)
- ○ Linux options, such as Apache HTTP Web Server, Nginx, and Apache Tomcat

- To run an HTTP server on AWS, you must find a service that provides compute power in the AWS Management Console.
- **At a fundamental level, three types of compute options are available –**
  - ○ **Virtual machines (VMs):**
    - a virtual machine emulates a physical server and allows you to install an HTTP server to run your applications.
    - To run virtual machines, you install a hypervisor on a host machine.
    - The hypervisor provisions the resources to create and run your VMs.
    - In AWS, virtual machines are called **Amazon Elastic Compute Cloud, or Amazon EC2.**
    - AWS operates and manages the host machines and the hypervisor layer.
    - AWS also installs the virtual machine operating system, called the guest operating system.
  - ○ **container services:**
  - ○ **serverless:**

Comparison

Category	AWS service
Instances (virtual machines)	• **Amazon Elastic Compute Cloud** (Amazon EC2) — Secure and resizeable compute capacity (virtual servers) in the cloud • **Amazon EC2 Spot Instances** — Run fault-tolerant workloads for up to 90% off • **Amazon EC2 Auto Scaling** — Automatically add or remove compute capacity to meet changes in demand • **Amazon Lightsail** — Easy-to-use cloud platform that offers you everything you need to build an application or website • **AWS Batch** — Fully managed batch processing at any scale
Serverless	• **AWS Lambda** — Run code without thinking about servers. Pay only for the compute time you consume.
Edge and hybrid	• **AWS Outposts** — Run AWS infrastructure and services on premises for a truly consistent hybrid experience • **AWS Snow Family** — Collect and process data in rugged or disconnected edge environments • **AWS Wavelength** — Deliver ultra-low latency application for 5G devices • **VMware Cloud on AWS** — Preferred service for all vSphere workloads to rapidly extend and migrate to the cloud • **AWS Local Zones** — Run latency sensitive applications closer to end-users
Containers	• **Amazon Elastic Container Service (Amazon ECS)** — Highly secure, reliable, and scalable way to run containers • **Amazon ECS Anywhere** — Run containers on customer-managed infrastructure • **Amazon Elastic Container Registry** (Amazon ECR) — Easily store, manage, and deploy container images • **Amazon Elastic Kubernetes Service** (Amazon EKS) — Fully managed Kubernetes service • **Amazon EKS Anywhere** — Create and operate Kubernetes clusters on your own infrastructure • **AWS Fargate** — Serverless compute for containers • **AWS App Runner** — Build and run containerized applications on a fully managed service
Cost and capacity management	• **AWS Savings Plan** — Flexible pricing model that provides savings of up to 72% on AWS compute usage • **AWS Compute Optimizer** — Recommends optimal AWS compute resources for your workloads to reduce costs and improve performance • **AWS Elastic Beanstalk** — Easy-to-use service for deploying and scaling web applications and services • **EC2 Image Builder** — Build and maintain secure Linux or Windows Server images • **Elastic Load Balancing** (ELB) — Automatically distribute incoming application traffic across multiple targets

AWS Compute services:

Category	Service description	AWS service
Instances (virtual machines)	Secure and resizable compute capacity (virtual servers) in the cloud	**Amazon Elastic Compute Cloud (EC2)**
	Run fault-tolerant workloads for up to 90% off	**Amazon EC2 Spot**
	Automatically add or remove compute capacity to meet changes in demand	**Amazon EC2 Autoscaling**
	Easy-to-use cloud platform that offers you everything you need to build an application or website	**Amazon Lightsail**
	Fully managed batch processing at any scale	**AWS Batch**
Serverless	Run code without thinking about servers. Pay only for the compute time you consume	**AWS Lambda**
Edge and hybrid	Run AWS infrastructure and services on premises for a truly consistent hybrid experience	**AWS Outposts**
	Collect and process data in rugged or disconnected edge environments	**AWS Snow Family**
	Deliver ultra-low latency application for 5G devices	**AWS Wavelength**
	Preferred service for all vSphere workloads to rapidly extend and migrate to the cloud	**VMware Cloud on AWS**
	Run latency sensitive applications closer to end-users	**AWS Local Zones**
Containers	Highly secure, reliable, and scalable way to run containers	**Amazon Elastic Container Service (ECS)**
	Run containers on customer-managed infrastructure	**Amazon ECS Anywhere**
	Easily store, manage, and deploy container images	**Amazon Elastic Container Registry (ECR)**
	Fully managed Kubernetes service	**Amazon Elastic Kubernetes Service (EKS)**
	Create and operate Kubernetes clusters on your own infrastructure	**Amazon EKS Anywhere**
	Serverless compute for containers	**AWS Fargate**
	Build and run containerized applications on a fully managed service	**AWS App Runner**
Cost and capacity management	Flexible pricing model that provides savings of up to 72% on AWS compute usage	**AWS Savings Plan**
	Recommends optimal AWS compute resources for your workloads to reduce costs and improve performance	**AWS Compute Optimizer**
	Easy-to-use service for deploying and scaling web applications and services	**AWS Elastic Beanstalk**
	Build and maintain secure Linux or Windows Server images	**EC2 Image Builder**
	Automatically distribute incoming application traffic across multiple targets	**Elastic Load Balancing (ELB)**

The AWS Nitro System : The Nitro System is comprised of three main parts: The Nitro Cards are a family of cards that offloads and accelerates I/O for functions including Amazon Virtual Private Cloud (VPC), Amazon Elastic Block Store (EBS), and Amazon EC2 instance storage, ultimately increasing overall system performance. The Nitro Hypervisor is a lightweight hypervisor that manages memory and CPU allocation and delivers performance that is indistinguishable from bare metal.

>>>
Amazon Elastic Compute Cloud
TODO:https://www.youtube.com/playlist?list=PL2yQDdvlhXf9A3sGKpKO85s8TSsIjvB2d
>>>

Amazon EC2 : is a web service that provides secure, resizable compute capacity in the cloud.

- It allows you to provision virtual servers called EC2 instances.
- not limited to running just web servers on your EC2 instances.
- You can create and manage EC2 instances through
    - the AWS Management Console,
    - the AWS Command Line Interface (CLI),
    - AWS software development kits (SDKs),
    - automation tools, and
    - infrastructure orchestration services.
- To create an EC2 instance, you must define the following:
    - Hardware specifications, like CPU, memory, network, and storage
    - Logical configurations, like networking location, firewall rules, authentication, and the operating system of your choice.
    - operating system you want by selecting an Amazon Machine Image (AMI)
- Use cases :
    - Run cloud-native and enterprise applications
    - Scale for HPC applications
    - Develop for Apple platforms
    - Train and deploy ML applications

Amazon Machine Image : In the AWS Cloud, the operating system installation is not your responsibility. Instead, it's built into the AMI that you choose.

- you can select storage mappings, the architecture type (such as 32-bit, 64-bit, or 64-bit ARM), and additional software installed.
- AMIs is that they are reusable.
- Each AMI in the AWS Management Console has an AMI ID, which is prefixed by “ami-”, followed by a random hash of numbers and letters. The IDs are unique to each AWS Region.

EC2 Image Builder is a fully managed AWS service that helps you to automate the creation, management, and deployment of customized, secure, and up-to-date server images.

- You own the customized images that Image Builder creates in your account.
- You can configure pipelines to automate updates and system patching for the images that you own.
- You can also run a stand-alone command to create an image with the configuration resources that you've defined.
- EC2 Image Builder has two runtime stages: build and test. Each runtime stage has one or more phases with configuration defined by the component document.
- There is no cost to use EC2 Image Builder to create custom AMI or container images. However, standard pricing applies for other services that are used in the process.

NOTE: EC2 instances are live instantiations of what is defined in an AMI, much like a cake is a live instantiation of a cake recipe.

- much like relationship between a class and an object. A class is something you model and define, while an object is something you interact with.
- the AMI is how you model and define your instance, EC2 instance is the entity you interact with, where you can install your web server and serve your content to users.

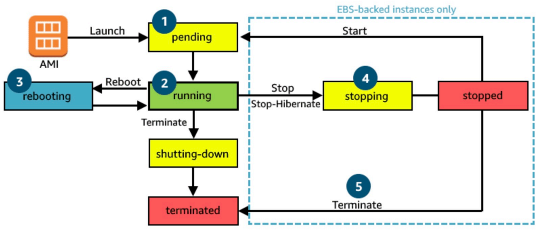
- **HOW IT WORKS:**
  - When you launch a new instance, AWS allocates a virtual machine that runs on a hypervisor.
  - Then, the AMI you selected is copied to the root device volume, which contains the image used to boot the volume.
  - In the end, you get a server that you can connect to and install packages and additional software on.

NOTE: An **Amazon Machine Image (AMI)** is the **basic unit of deployment in Amazon EC2**, and is one of the types of images you can create with Image Builder
An image pipeline provides an automation framework for building secure AMIs and container images on AWS.
A managed image is a resource in Image Builder that consists of an AMI or container image, plus metadata, such as version and platform.
An Image Builder image recipe is a document that defines the base image and the components that are applied to the base image to produce the desired configuration **for the output AMI image.**
An Image Builder container recipe is a document that defines the base image and the components that are applied to the base image to produce the desired configuration **for the output container image.**
The base image is the selected image and operating system used in your image or container recipe document, along with the components. The base image and the component definitions combined produce the desired configuration for the output image.
A component defines the sequence of steps required to either customize an instance prior to image creation (a build component), or to test an instance that was launched from the created image (a test component).

Instance Family	Description	Use Cases
General purpose	Provides a balance of compute, memory, and networking resources, and can be used for a variety of workloads.	Scale out workloads, such as web servers, containerized microservices, caching fleets, distributed data stores, and development environments
Compute optimized	Ideal for compute-bound applications that benefit from high-performance processors.	High-performance web servers, scientific modeling, batch processing, distributed analytics, high-performance computing (HPC), machine/deep learning, ad serving, highly scalable multiplayer gaming
Memory optimized	Designed to deliver fast performance for workloads that process large datasets in memory.	Memory-intensive applications, such as high-performance databases, distributed web-scale in-memory caches, mid-size in-memory databases, real-time big-data analytics, and other enterprise applications
Accelerated computing	Use hardware accelerators or co-processors to perform functions such as floating-point number calculations, graphics processing, or data pattern matching more efficiently than is possible with conventional CPUs.	3D visualizations, graphics-intensive remote workstations, 3D rendering, application streaming, video encoding, and other server-side graphics workloads
Storage optimized	Designed for workloads that require high, sequential read and write access to large datasets on local storage. They are optimized to deliver tens of thousands of low-latency random I/O operations per second (IOPS) to applications that replicate their data across different instances.	NoSQL databases, such as Cassandra, MongoDB, and Redis, in-memory databases, scale-out transactional databases, data warehousing, Elasticsearch, and analytics

1. Amazon EC2 instances are a combination of virtual processors (vCPUs), memory, network, and, in some cases, instance storage and graphics processing units (GPUs). When you create an EC2 instance, you need to choose how much you need of each of these components.
2. By default, your EC2 instances are placed in a network called the default Amazon Virtual Private Cloud (Amazon VPC).
3. Any resource you put inside the default VPC will be public and accessible by the internet, so you shouldn't place any customer data or private information in it.
4. When architecting any application for high availability, consider using at least two EC2 instances in two separate Availability Zones.

>>>
Amazon EC2 Instance Lifecycle
>>



1. When you launch an instance, it enters the pending state. **When an instance is pending, billing has not started.** At this stage, the instance is preparing to enter the running state. Pending is where AWS performs all actions needed to set up an instance, such as copying the AMI content to the root device and allocating the necessary networking components.
2. When your instance is running, it's ready to use. This is also the stage where **billing begins**. As soon as an instance is running, you can take other actions on the instance, such as reboot, terminate, stop, and stop-hibernate.
3. When you reboot an instance, it's different than performing a stop action and then a start action. Rebooting an instance is equivalent to rebooting an operating system. The instance remains on the same host computer, and maintains its public and private IP address, in addition to any data on its instance store.
4. It typically takes a few minutes for the reboot to complete. When you stop and start an instance, your instance may be placed on a new underlying physical server. Therefore, you lose any data on the instance store that were on the previous host computer. **When you stop an instance, the instance gets a new public IP address but maintains the same private IP address.**
5. **When you terminate an instance, the instance stores are erased, and you lose both the public IP address and private IP address of the machine.** Termination of an instance means that you can no longer access the machine.

NOTE: When you stop an instance, it enters the stopping state until it reaches the stopped state. AWS does not charge usage or data transfer fees for your instance after you stop it, but storage for any Amazon EBS volumes is still charged. While your instance is in the stopped state, you can modify some attributes, like the instance type. When you stop your instance, the data stored in memory (RAM) is lost.

NOTE: When you stop-hibernate an instance, AWS signals the operating system to perform hibernation (suspend-to-disk), which saves the contents from the instance memory (RAM) to the Amazon EBS root volume.

USE CASE:
Consider a scenario where you build a standard three-tier application, where you have web servers, application servers, and database servers. Suddenly, the application you built becomes extremely popular. To relieve some stress on the database that supports your application, you want to implement a custom backend layer that caches database information in memory (RAM). You decide to run this custom backend caching solution on Amazon EC2.

In this scenario, the stop-hibernate feature would be instrumental in persisting storage. It would prevent you from having to manually create scripts to save the RAM data before shutting down the server.

AWS offers three main purchasing options for EC2 instances

– On-Demand : Pay as you go

: Billing begins whenever the instance is running, and billing stops when the instance is in a stopped or terminated state.

: The price per second for a running On-Demand Instance is fixed.

- Reserved : Reserve capacity with Reserved Instances (RIs)

: RIs provide a discounted hourly rate and an optional capacity reservation for EC2 instances.

: You can choose between three payment options – All Upfront, Partial Upfront, or No Upfront.

: You can select either a 1-year or 3-year term for each of these options.

: When you stop an RI, you still pay for it because you committed to a 1-year or 3-year term.

: Discounted differently - The discount applied by a Reserved Instance purchase is not directly associated with a specific instance ID, but with an instance type.

- All Upfront offers a higher discount than Partial Upfront instances.
- Partial Upfront instances offer a higher discount than No Upfront.
- No Upfront offers a higher discount than On-Demand.

- Spot Instances : Save on costs

: Amazon EC2 Spot Instances allow you to take advantage of unused EC2 capacity in the AWS Cloud.

: They are available at up to a 90% discount compared to On-Demand prices.

: With Spot Instances, you set a limit on how much you would like to pay for the instance hour.

: This is compared against the current Spot price that AWS determines.

- If the amount you pay is more than the current Spot price and there is capacity, then you will receive an instance.

: if AWS determines that capacity is no longer available for a particular Spot Instance or if the Spot price exceeds how much you are willing to pay, AWS will give you a 2-minute warning before it interrupts your instance. That means any application or workload that runs on a Spot Instance must be able to be interrupted so inherently fault-tolerant workloads are typically good candidates to use with Spot Instances. Example: big data, containerized workloads, continuous integration/continuous delivery (CI/CD), web servers, high-performance computing (HPC), image and media rendering, and other test and development workloads.

>>

Launch a web application on Amazon EC2

>>

AWS account ID : 213629629125

Account name : Nanditha_AWSAccount

Email address : nanditha.murthy@gmail.com

Canonical user ID : 6a593b821b8a31babd0395bdcaf1cd678449ce8d7f8ad70eab7fd70a3a25a3f8

1. **Launch an Amazon EC2 Instance**
2. **Adding instructions to user data**
3. **Test web application**
4. **Connect to EC2 instance console**

>>
Container Services
>>

Manage containers with **Amazon Elastic Container Service (Amazon ECS)**

- Amazon ECS is an end-to-end container orchestration service that helps you spin up new containers and manage them across a cluster of EC2 instances.
- To run and manage your containers, you need to install the Amazon ECS container agent on your EC2 instances.
- Containers share the same operating system and kernel as the host they exist on, whereas virtual machines contain their own operating system. Each virtual machine must maintain a copy of an operating system, which results in a degree of wasted resources.

Use Kubernetes with **Amazon Elastic Kubernetes Service (Amazon EKS)**

- Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services.
- An EC2 instance with the ECS agent installed and configured is called a container instance. In Amazon EKS, it is called a worker node.
- An ECS container is called a task. In Amazon EKS, it is called a pod.
- While Amazon ECS runs on AWS native technology, Amazon EKS runs on top of Kubernetes.

NOTE: If you run your code on Amazon EC2, AWS is responsible for the physical hardware, and you are responsible for the logical controls, such as guest operating system, security and patching, networking, security, and scaling.

NOTE: If you run your code in containers on Amazon ECS and Amazon EKS, AWS is responsible for more of the container management, such as deploying containers across EC2 instances and managing the container cluster. However, when running ECS and EKS on EC2, you are still responsible for maintaining the underlying EC2 instances.

NOTE: If you want to deploy your workloads and applications without having to manage any EC2 instances, you can do that on AWS with serverless compute. (Ex: AWS Fargate and AWS Lambda)

AWS Fargate is a purpose-built serverless compute engine for containers. Fargate scales and manages the infrastructure, allowing developers to work on what they do best – application development.

With **AWS Lambda,** you can run code without provisioning or managing servers or containers. You can run code for virtually any type of application or backend service, including data processing, real-time stream processing, machine learning, WebSockets, IoT backends, mobile backends, and web apps, like your corporate directory app! requires zero administration from the user.

- A Lambda function has three primary components – trigger, code, and configuration.
- You are charged for the number of times your code is triggered (requests) and for the time your code executes, rounded up to the nearest 1 ms (duration).

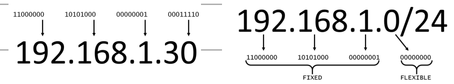
>>

NETWORKING

>>

IP address in its binary format. Instead, it's converted into decimal format and noted as an Ipv4 address.

- the 32 bits are grouped into groups of 8 bits, also called octets. Each of these groups is converted into decimal format separated by a period.

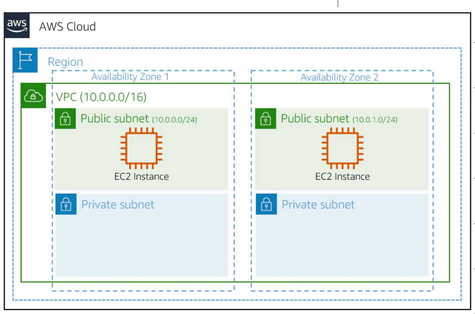


Classless Inter-Domain Routing (CIDR) notation : CIDR notation is a compressed way of specifying a range of IP addresses.

- It begins with a starting IP address and is separated by a forward slash (the "/" character) followed by a number. The number at the end specifies how many of the bits of the IP address are fixed.
- 32 total bits subtracted by 24 fixed bits leaves 8 flexible bits. Each of these flexible bits can be either 0 or 1, because they are binary. That means that you have two choices for each of the 8 bits, providing 256 IP addresses in that IP range.
- In AWS, the smallest IP range you can have is /28, which provides 16 IP addresses. The largest IP range you can have is a /16, which provides 65,536 IP addresses.

Amazon Virtual Private Cloud :

- A **virtual private cloud (VPC)** is an isolated network that you create in the AWS Cloud, similar to a traditional network in a data center.
- When you create a VPC, you must choose three main factors:

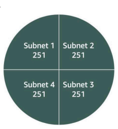


  - Name of the VPC.
  - Region where the VPC will live. Each VPC spans multiple Availability Zones within the selected Region.
  - IP range for the VPC in CIDR notation. This determines the size of your network. Each VPC can have up to four /16 IP ranges.
- Think of **subnets** as smaller networks inside your base network – or virtual local area networks (VLANs) in a traditional,
- When you create a subnet, you must specify the following:
  - VPC you want your subnet to live in. In this case: VPC (10.0.0.0/16)
  - Availability Zone you want your subnet to live in. In this case: AZ1
  - CIDR block for your subnet, which must be a subset of the VPC CIDR block. In this case: 10.0.0.0/24

When you launch an EC2 instance, you launch it inside a subnet, which will be located inside the Availability Zone you choose.

Reserved IP: For AWS to configure your VPC appropriately, AWS reserves five IP addresses in each subnet. These IP addresses are used for routing, Domain Name System (DNS), and network management.

Internet gateway : To enable internet connectivity for your VPC, you must create an internet gateway. Just as a modem connects your computer to the internet, the internet gateway connects your VPC to the internet.

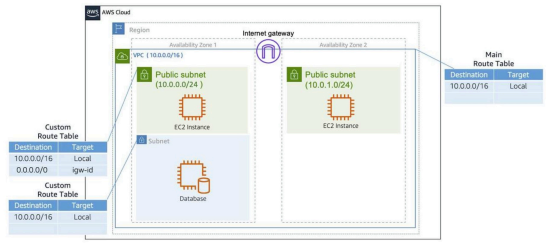


IP address	Reserved for
10.0.0.0	Network address
10.0.0.1	VPC local router
10.0.0.2	DNS server
10.0.0.3	Future use
10.0.3.255	Network broadcast address

A virtual private gateway connects your AWS VPC to another private network.

- Once you create and attach a virtual private gateway to a VPC, the gateway acts as anchor on the AWS side of the connection.
- On the other side of the connection, you will need to connect a customer gateway to the other private network. A customer gateway device is a physical device or software application on your side of the connection.Once you have both gateways, you can then establish an encrypted VPN connection between the two sides.

Amazon VPC Routing:



- When you create a VPC, AWS creates a route table called the **main route table.**
  - A route table contains a set of rules, called routes, that are used to determine where network traffic is directed.
  - AWS assumes that when you create a new VPC with subnets, you want traffic to flow between them. Therefore, the default configuration of the main route table is to allow traffic between all subnets in the local network.
- The destination and target are two main parts of this route table.
  - The destination is a range of IP addresses where you want your traffic to go. In this case, the destination is the VPC network's IP range.
  - The target is the connection through which to send the traffic. In this case, the traffic is routed through the local VPC network.

Custom route tables: - While the main route table is used implicitly by subnets that do not have an explicit route table association, you might want to provide different routes on a per-subnet basis, for traffic to access resources outside of the VPC.

- If you associate a custom route table with a subnet, the subnet will use it instead of the main route table.
- Each custom route table you create will have the local route already inside it, allowing communication to flow between all resources and subnets inside the VPC.
- The local route cannot be deleted.

Amazon VPC Security: - network access control list (network ACL) as a firewall at the subnet level.

- A **network ACL** enables you to control what kind of traffic is allowed to enter or leave your subnet.
- Network ACLs are considered stateless, so you need to include both the inbound and outbound ports used for the protocol.
- If you don't include the outbound range, your server would respond but the traffic would never leave the subnet.

Secure EC2 instances with security groups:

- **a firewall called a security group.** The default configuration of a security group blocks all inbound traffic and allows all outbound traffic.
- security groups are stateful.
- If you want your EC2 instance to accept traffic from the internet, you must open up inbound ports.

>>>
Storage Types
>>

- **Block storage :**
  - block storage splits files into fixed-size chunks of data called blocks that have their own addresses
  - Block storage in the cloud is analogous to direct-attached storage (DAS) or a storage area network (SAN).
  - are fast and use less bandwidth.
  - optimized for low-latency operations
  - Use cases:
    - high-performance enterprise workloads, such as databases or enterprise resource planning (ERP) systems, that require low-latency storage.
- **Object storage :**
  - a single unit of data when stored
  - these objects are stored in a flat structure instead of a hierarchy.
  - Each object is a file with a unique identifier. This identifier, along with any additional metadata, is bundled with the data and stored.
  - Use cases:
    - large datasets; unstructured files, like media assets; and static assets, like photos.
- **File storage :**
  - file storage treats files as a singular unit
  - Ideal when you require centralized access to files that need to be easily shared and managed by multiple host computers.
  - this storage is mounted onto multiple hosts, and requires file locking and integration with existing file system communication protocols.
  - File storage systems are often supported with a network attached storage (NAS) server.
  - Use cases:
    - Large content repositories
    - Development environments
    - User home directories

Amazon EC2 Instance Storage :

- Amazon EC2 instance store provides temporary block-level storage for an instance.
- This storage is located on disks that are physically attached to the host computer.
- This ties the lifecycle of the data to the lifecycle of the EC2 instance.

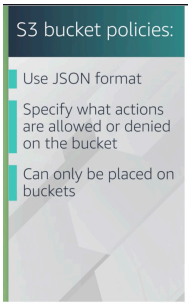
Amazon Elastic Block Store :

- Amazon EBS is a block-level storage device that you can attach to an Amazon EC2 instance.
- EBS volumes are essentially drives of a user-configured size attached to an EC2 instance, similar to how you might attach an external drive to your laptop.
- You can detach an EBS volume from one EC2 instance and attach it to another EC2 instance in the same Availability Zone, to access the data on it.
- You can scale Amazon EBS volumes in two ways.
  - *Increase the volume size, as long as it doesn't increase above the maximum size limit*: For EBS volumes, the maximum amount of storage you can have is 16 TB.
  - *Attach multiple volumes to a single Amazon EC2 instance*. EC2 has a one-to-many relationship with EBS volumes.
- Amazon EBS benefits:
  - High availability
  - Data persistence
  - Data encryption
  - Flexibility
  - Backups
- Amazon EBS volumes are organized into two main categories –

- solid-state drives (SSDs) : provide strong performance for random input/output (I/O),
- hard-disk drives (HDDs) : provide strong performance for sequential I/O.

Volume Types	Description	Use Cases	Volume Size	Max IOPS	Max Throughput
EBS Provisioned IOPS SSD	Highest performance SSD designed for latency-sensitive transactional workloads	I/O-intensive NoSQL and relational databases	4 GB–16 TB	64,000	1,000 MB/s
EBS General Purpose SSD	General purpose SSD that balances price and performance for a wide variety of transactional workloads	Boot volumes, low-latency interactive apps, development, and test	1 GB–16 TB	16,000	250 MB/s
Throughput Optimized HDD	Low-cost HDD designed for frequently accessed, throughput-intensive workloads	Big data, data warehouses, log processing	500 GB–16 TB	500	500 MB/s
Cold HDD	Lowest cost HDD designed for less frequently accessed workloads	Colder data requiring fewer scans per day	500 GB–16 TB	250	250 MB/s

Object Storage with Amazon Simple Storage Service:



- Amazon Simple Storage Service (Amazon S3) is a standalone storage solution that isn't tied to compute.
- Amazon S3 is an object storage service.
- Object storage stores data in a flat structure, using unique identifiers to look up objects when requested.
- An object is a file combined with metadata.
- you store your objects in containers called buckets. You can't upload any object, not even a single photo, to Amazon S3 without creating a bucket first.
- When you choose a Region for your bucket, all objects you put inside the bucket will be redundantly stored across multiple devices, across multiple Availability Zones.

- **Amazon S3 use cases**
  - Backup and storage
  - Media hosting
  - Software delivery
  - Data lakes
  - Static websites
  - Static content
- Amazon S3 bucket policies are defined in a JSON format.
  - The difference is IAM policies are attached to users, groups, and roles, whereas S3 bucket policies are only attached to S3 buckets.
  - S3 bucket policies specify what actions are allowed or denied on the bucket.
  - S3 bucket policies have a larger size limit.
  - S3 bucket policies can only be placed on buckets, and cannot be used for folders or objects. However, the policy that is placed on the bucket applies to every object in that bucket.

Amazon Elastic File System (Amazon EFS) and Amazon FSx:

- For file storage that can mount on to multiple EC2 instances, you can use Amazon Elastic File System (Amazon EFS) or Amazon FSx.

Service	Description	FAQs
Amazon Elastic File System (Amazon EFS)	Fully managed NFS file system	EFS FAQs
Amazon FSx for Windows File Server	Fully managed file server built on Windows Server that supports the SMB protocol	FSx for Windows File Server FAQs
Amazon FSx for Lustre	Fully managed Lustre file system that integrates with S3	FSx for Lustre FAQs

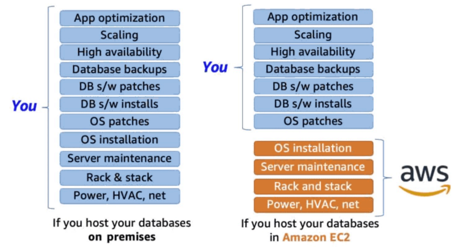
- It is file storage.
- You pay for what you use (you don't have to provision storage in advance).
- Amazon EFS and Amazon FSx can be mounted onto multiple EC2 instances.

>>

Database

>>

Unmanaged database: Managed database:



Database Type	Use Cases	AWS Service
Relational	Traditional applications, ERP, CRM, e-commerce	Amazon RDS, Amazon Aurora, Amazon Redshift
Key-value	High-traffic web apps, e-commerce systems, gaming applications	Amazon DynamoDB
In-memory	Caching, session management, gaming leaderboards, geospatial applications	Amazon ElastiCache for Memcached, Amazon ElastiCache for Redis
Document	Content management, catalogs, user profiles	Amazon DocumentDB (with MongoDB compatibility)
Wide column	High-scale industrial apps for equipment maintenance, fleet management, and route optimization	Amazon Keyspaces (for Apache Cassandra)
Graph	Fraud detection, social networking, recommendation engines	Amazon Neptune
Time series	IoT applications, DevOps, industrial telemetry	Amazon Timestream
Ledger	Systems of record, supply chain, registrations, banking transactions	Amazon QLDB

Amazon Relational Database Service

- Amazon Relational Database Service (Amazon RDS) lets customers create and manage relational databases in the cloud without the operational burden of traditional database management.
- Amazon RDS engines are:
  - Commercial: Oracle, SQL Server
  - Open Source: MySQL, PostgreSQL, MariaDB
  - Cloud Native: Amazon Aurora : The cloud native option, Amazon Aurora, is a MySQL- and PostgreSQL-compatible database built for the cloud.
- The compute portion is called the DB (database) instance, which runs the database engine.
- A DB instance can contain multiple databases with the same engine, and each database can contain multiple tables.
- supports three instances:
  - Standard, which includes general-purpose instances
  - Memory Optimized, which is optimized for memory-intensive applications
  - Burstable Performance, which provides a baseline performance level, with the ability to burst to full CPU usage
- a DB instance uses Amazon Elastic Block Store (EBS) volumes as its storage layer.
- Amazon EBS volume storage types:
  - General purpose (SSD)
  - Provisioned IOPS (SSD)
  - Magnetic storage (not recommended)

Amazon DynamoDB

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- You can scale up or scale down your tables' throughput capacity without downtime or performance degradation.
- You can use the AWS Management Console to monitor resource usage and performance metrics.
- DynamoDB automatically spreads the data and traffic for your tables over a sufficient number of servers to handle your throughput and storage requirements, while maintaining consistent and fast performance.
- DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.
- basic DynamoDB components:
  - Tables : A table is a collection of data.
  - Items : An item is a group of attributes that is uniquely identifiable among all the other items.
  - Attributes : An attribute is a fundamental data element, something that does not need to be broken down any further.

>>>
Application Management
>>>

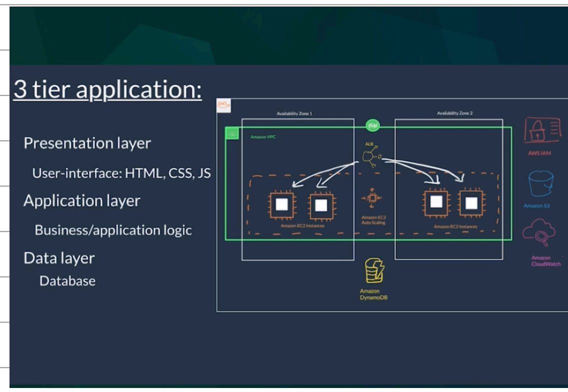
Monitoring using Amazon CloudWatch: You can use CloudWatch to:

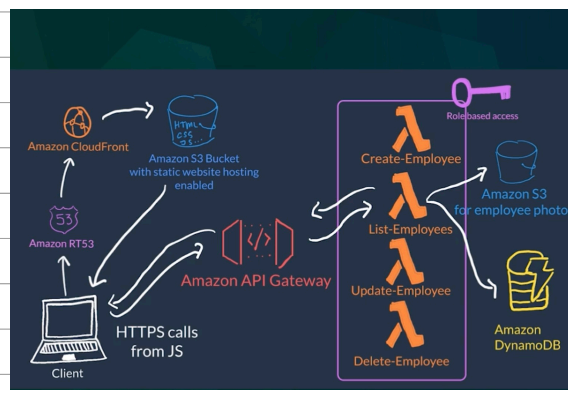
- Detect anomalous behavior in your environments
- Set alarms to alert you when something is not right
- Visualize logs and metrics with the AWS Management Console
- Take automated actions like scaling
- Troubleshoot issues
- Discover insights to keep your applications healthy

Traffic Routing with Amazon Elastic Load Balancing

- Load balancing refers to the process of distributing tasks across a set of resources.
- To do this, you first need to enable the load balancer to take all of the traffic and redirect it to the backend servers based on an algorithm.
- The request is sent to a load balancer. Then, it's sent to one of the EC2 instances that hosts the application. The return traffic goes back through the load balancer and back to the client's browser. As you can see, the load balancer is directly in the path of the traffic.
-

Feature	Application Load Balancer	Network Load Balancer
Protocols	HTTP, HTTPS	TCP, UDP, TLS
Connection draining (deregistration delay)	√	√
IP addresses as targets	√	√
Static IP and Elastic IP address		√
Preserve Source IP address		√
Routing based on Source IP address, path, host, HTTP headers, HTTP method, and query string	√	
Redirects	√	
Fixed response	√	
User authentication	√	

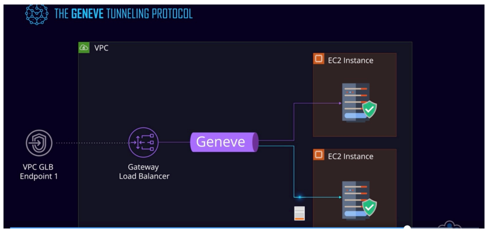




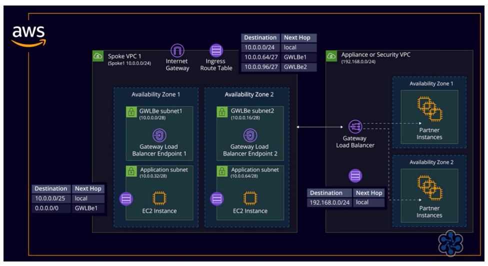
>>>

The Gateway Load Balancer

>>>



- The Gateway Load Balancer consists of two parts.
  - **VPC Gateway Load Balancer Endpoint:** This endpoint is expected to be defined in the VPC where you want to protect the traffic.
  - **actual Gateway Load Balancer:** sends traffic to a fleet of EC2 instances running third party network appliance software.
- **GENEVE** is a tunneling mechanism which provides extensibility while still using the offload capabilities of Network Interface Cards for performance improvement. GENEVE works by creating a Layer 2 logical network that is encapsulated in UDP packets.



- A tunnel is created between the Gateway Load Balancer and the fleet of instances on the back-end.
- Traffic is encapsulated and sent through the tunnel to the security appliances implemented in EC2 instances, which will examine and act on packets as they're sent or received.
- The Gateway Load Balancer encapsulates the packets to the target to provide separation and add some additional information about which Gateway Load Balancer Endpoint the packet came from. GENEVE uses port 6081 to get traffic from the Gateway Load Balancer, and it uses HTTP port 80 for health checks.
- Gateway Load Balancer endpoints can be added to a route table as the next hop and integrate the Gateway Load Balancer into the traffic flow of a VPC.
- In the general process of setting up a Gateway Load Balancer, you need to provision a VPC dedicated to the Gateway Load Balancer and the third party virtual appliance software you're running on EC2 instances.
- On the VPC where your application lives, you create Gateway Load Balancer endpoints on their own dedicated subnets and update the route tables to include the Gateway Load Balancer endpoints for traffic coming from your application subnets to them and traffic from the Internet Gateway to them as well, in order to integrate the security VPC to the traffic flow.
- steps are:
  - Step 1, locate the partner's virtual appliance software, perhaps in AWS Marketplace.
  - Step 2, launch the appliance instances in your security VPC.
  - Step 3, create a Gateway Load Balancer and target group with those appliance instances.
  - Step 4, create Gateway Load Balancer endpoints in the VPC where the traffic needs to be inspected.
  - step 5, update route tables to make Internet Gateway move traffic to and from the Gateway endpoints and the Gateway Load Balancer endpoint as the next-hop.

- **General architecture diagram**

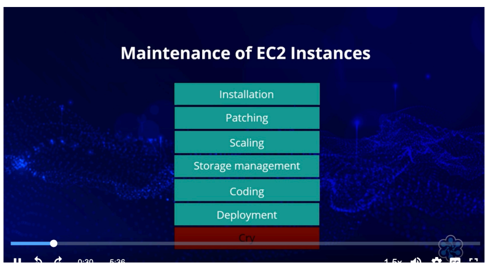
- we have an application deployed to private subnets in an auto-scaling target group service by an Application Load Balancer. The Application Load Balancers are deployed to public subnets. We also have a separate security VPC with a Gateway Load Balancer and security appliances in a target group for auto-scaling. This will allow for the appliance fleet to adjust based on application load, and therefore, scaling horizontally.
- a packet/Traffic flow will travel in this architecture:
  - **Step 1,** a customer access your web application and a request is generated.
  - **Step 2,** the landing place for public traffic request is the Application Load Balancer. However, the Internet Gateway is configured with an Ingress route table to direct traffic to a Gateway Load Balancer endpoint.
  - **In step 3**, the Gateway Load Balancer endpoint directs traffic to the Gateway Load Balancer in the security VPC.
  - **Step 4,** at the Gateway Load Balancer, the packets are wrapped using the GENEVE tunneling protocol and dispatch through the security appliance selected.
  - **Step 5,** the packet analysis takes place in the security appliance. What actually happens depends on the appliance being used and the configuration that you define.
  - **Step 6,** after analysis, the packets are sent back, still encapsulated to the Gateway Load Balancer where the encapsulation is removed and traffic is sent to the Gateway Load Balancer endpoint where it originally came from.
  - **Step 7,** the corresponding Gateway Load Balancer endpoint will direct traffic to the Application Load Balancer and will target your application.
  - **From step 8,** the response flow is very similar in that the application response will pass through the Application Load Balancer and into the subnet with the Gateway Load Balancer endpoint in the same availability zone. This will send traffic to the Gateway Load Balancer yet again in the return path for the security VPC.

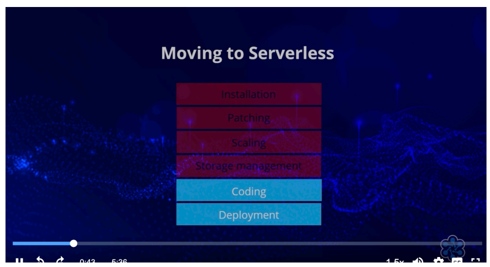
- **Step 9,** the packets are encapsulated yet again and sent to the security appliance where they are processed.
- **Step 10,** the packets are sent back to the Gateway Load Balancer where encapsulation is removed and packets moved to the Gateway Load Balancer endpoint.
- **Step 11,** the Gateway Load Balancer endpoint will send traffic out through the Internet Gateway.
- **step 12,** the response is received by the customer. So, as we get to see from this example flow, the Gateway Load Balancer allows you to leverage and horizontally scale third-party security appliances from the AWS Marketplace in your Amazon VPCs.

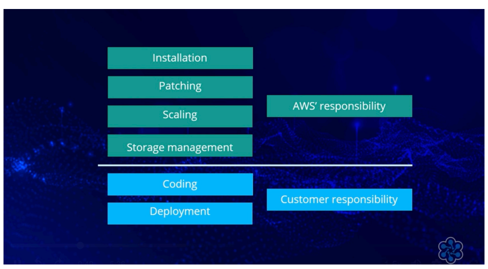
>>
AWS Outposts
>>

- hybrid functionality, helping your align your applications and infrastructure from your on-premises environment with that of the AWS cloud.
- With AWS Outposts, it's now possible to bring the AWS cloud to your data center.
- There are two different options available when using Outposts.
  - You can either use VMware on AWS, which will seamlessly run your existing VMware management and infrastructure, or
  - you can use a native AWS variant, which means you can use the same APIs and management tools as you would in AWS but on premises.
- By bringing the same native AWS tools, API, management console and services to your on premises data center, it enables you to manage and implement a hybrid cloud approach with seamless migration abilities, burst capacity and management between local an AWS cloud environments.
- AWS Outposts is that the service itself is fully managed. This means that you do not need to maintain a level of patch management across your infrastructure. AWS will ensure the infrastructure is patched as and when needed.

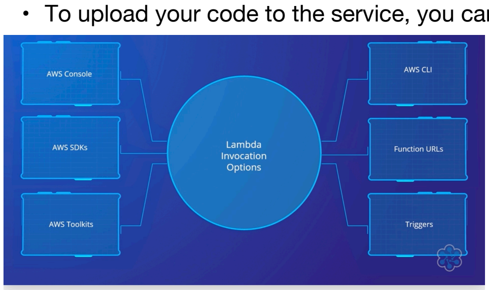
>>
AWS Lambda
>>

- 



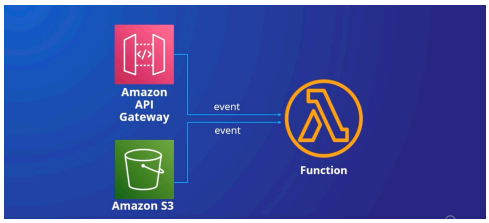


- There are three major parts:
  - The first piece is the input
  - The second piece is the function, and
  - The last piece is the output.
- Just as EC2 is made up of instances, Lambda is made up of functions. Functions are the code that you write that represents your business logic.
- The way you specify power is by choosing how much memory you want to allocate to your function. The service then uses this number and provides proportional amounts of CPU, network and disk I/O.
- To upload your code to the service, you can either write the code directly in the service itself or you can upload this code via a zip file or files stored in Amazon S3.



- The programming language you write your code in must match the runtime you select in the service.
- There are several options for your function to be **invoked:**
  - Your function could be invoked directly through the console, SDK, AWS toolkits, or through the CLI.
  - It could be invoked using a function URL, which is an HTTP endpoint you can enable for your Lambda function
  - Or it could be invoked automatically by a trigger, such as an AWS

service or resource. These triggers will run your function in response to certain events or on a schedule that you specify.



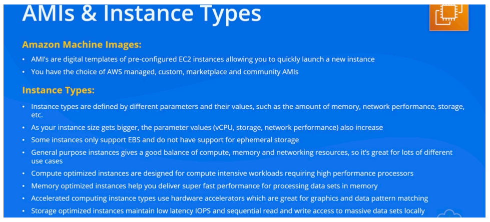
- No matter how you invoke a Lambda function, even if you're invoking it directly through the Lambda service itself, you're using the service's API. **Every invocation goes through the Lambda API.** What this API provides to you is three different models of how you can invoke your function:
  - **the synchronous or push-based model**. This follows the request/response model. For synchronous invocations, if the function fails, the trigger is responsible for retrying it. In some cases, this might mean that there are no retries.
  - **asynchronous model, also called the event-based model.** In this configuration, the response does not go back to the original service that invoked the lambda function. In fact, there's no path back up to the service that triggered the function to run, unless you write that logic yourself. This is common with Amazon S3 and Amazon Simple Notification Service. it handles retries if the function returns an error or is throttled. It also uses a built-in queue.
  - If any of these events can't be processed for whatever reason, you can **send that failed event to a dead letter queue or use Lambda destinations to send a record of the invocation to a service.**
    - The dead letter queue will receive only the content of the event while using Lambda destinations records will include both the request context and payload as well as the response context and payload.
    - While both are a great way to troubleshoot failed events, Destinations is the more feature-rich option.
  - **stream model, also called the poll-based model:** Typically used when you need to poll messages out of stream or queue-based services such as DynamoDB streams, Kinesis streams, and Amazon SQS.
    - the Lambda service runs a poll-er on your behalf, and consumes the messages and data that comes out of them, filtering through them to invoke your Lambda function on only messages that match your use case.
    - With this model, you would need to create an event source mapping to process items from your stream or queue.
    - An event source mapping links your event source to your Lambda function so that the events generated from your event source will invoke your function.
    - These mappings, the response you get back, the permissions you set up, the polling behavior and even the event itself, can be very different based on the event source you're using. However, the way you create event source mappings stays the same. You can do this by using the SDK or CLI.
- Now - keep in mind that if an AWS service invokes your function, the ability to select an invocation type is removed. The service gets this choice instead and selects the invocation method for you. So all of these hard choices go away.
- When you invoke your function, you can pass in events for the function to process. If a service invokes your function, they can also pass in events - however, the service will be responsible for structuring those events.
- example, your code could run in response to a request from API Gateway or an S3 event, such as a PUT object API call. So once the PUT object API call is made, AWS Lambda will run your code, just as you've written it, using only the compute power you defined.
- Output: Once the function is triggered, and your code runs, your Lambda function can then make calls to downstream resources. This means that from your code, you can make API calls to other services like Amazon DynamoDB, Amazon SQS, Amazon SNS and more.
- These log streams act as a recording of the sequence of events from your function.
- Lambda also sends common metrics of your functions to CloudWatch for monitoring and alerting.
- **Billing and payment:**
  - 1. You are charged for the amount of requests that you send to your function.
  - 2. The Lambda function begins charging you when it is triggered, and stops charging you when the code has been executed. Otherwise known as the duration it runs. This is rounded up to the nearest 1 millisecond of use.
  - 3. You're charged based on the amount of compute power you provision for your function. So if you provision the maximum amount of memory, you'll be charged for that.
- **monitoring and troubleshooting with Lambda**
- There are three main categories of metrics:
  - Invocation metrics
    - there is a metric called invocations that tracks the number of times the function has been invoked.
    - errors metric which counts the number of failed invocations of the function.
  - Performance metrics
    - provide performance details about an invocation. These metrics include duration, which measures how long the function runs in milliseconds from when it is invoked until it terminates.
    - IteratorAge metric which is only used for stream-based invocations such as Amazon Kinesis. It measures in time how long Lambda took to receive a batch of records to the time of the last record written to the stream. This IteratorAge is measured in milliseconds.
  - And concurrency metrics
    - When you want to monitor metrics for concurrency, you can use concurrent executions metric, which is a combined metric for all of your Lambda functions that you have running within your AWS account,
    - in addition to functions with a custom concurrency limit. It calculates the total sum of concurrent executions at any point in time.
  - CloudWatch : also gathers log data sent by Lambda to help you better troubleshoot and understand issues. For each function that you have running, CloudWatch will create a different log group. This is language dependent

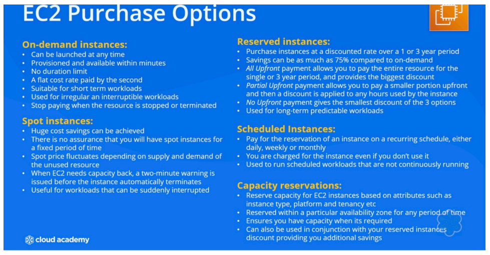
>>>

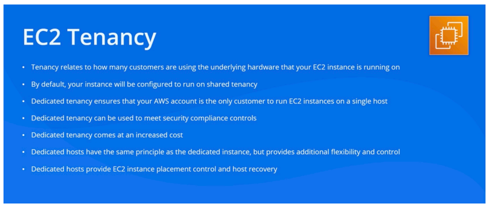
AWS Compute Summary Key Points

>>

- So you will be expected to know what comes with the AMI and what doesn't?







-
- you need to select when creating your instance for the exam, you should be aware of the different options that they offer such as the operating system that you'll be running in addition to any other additional software
- Some hands on experience with it and EC2 - help you to establish familiarity with the different steps involved in creating an instance.
- You need to be aware of the different instance types that are available.
- how the compute power and performance values fluctuate with instant size.
- answer any questions relating to EC2 workload efficiency.
- best instance purchase option to help you optimize the cost of your environment.
- you know the difference between on-demand, spot and reserved instances. Now depending on the scenario you will have to demonstrate your understanding of these different purchase options to help you determine under which circumstance you should use each of them.
    - If the question talks about how your workload is predictable and will be required for perhaps one or three years and you need to optimize costs, then reserved instances should come to mind and would likely be the answer.
    - If the question highlights how the workload can be interrupted. And again you're looking to build a cost efficient solution. Then this would be a good use case of spot instances.
- review the key differences of the purchase options and understand their specific use cases to help you optimize costs.
- Tenancy options of your instances
- Now by default, our instances run on shared tenancy. Whereby we share the underlying host with other customers.
- However, you might receive questions explaining that you need to secure your infrastructure to maintain compliance and ensure that your EC2 instances do not share any underlying host with any other customer. So how could you do that? Well, the answer would fall under your tenancy options. Either dedicated instances or dedicated hosts would resolve this issue. Now with dedicated hosts, it provides additional control over the placement of your EC2 instances on those hosts. So ensure you have a good understanding of your options here.
- how to automatically run commands on the first boot cycle of your instance.
- For example, you might need to perform operating system updates or install additional software from a repository when your instance first boots up. So how would you achieve this? Well, the answer lies in the user data section of your instance during its configuration. It allows you to enter commands to do exactly that. Also on this point, you can also use metadata of the instance to see the user data configuration for that instance. And this can be found by going to 169.254.169.254/latest/meta-data.
- key pairs could be one topic
- Ensure you are familiar with how to connect to both Windows and Linux based instances. Now Windows uses RDP on port 3389 and Linux uses SSH, which is on port 22.
- expected to know the main function of the service and the benefits it brings such as the ability to automatically increase or decrease your EC2 resources to meet the demands of your applications.
- to implement an efficient way to enhance the performance of the application after users complained of poor response. Now this might be caused by a bottleneck in your EC2 resources not being able to handle and process the amount of traffic. Now by implementing auto scaling, you could automatically increase your ET2 fleet size. Thereby you would increase the amount of resources and remove the bottleneck.
- assessed on your ability to optimize the cost of your EC2 fleet. Now, one way would be to remove unused resources. By implementing auto scaling, you can scale in your EC2 fleet by terminating unused capacity based on set thresholds
- So auto scaling is all about optimizing performance and cost. So look out for this as an option whenever you receive a question covering this topic. Now you will likely see questions with auto scaling interlinking with elastic load balances as well.
- And they work very well together. Elastic load balances allow you to manage loads across your target groups. Whereas EC2 auto scaling allows you to elastically scale those target groups based upon the demand.
- differentiate between auto-scaling and ELBs. Also, ensure you are familiar with the different ELBs that exist as you'll be assessed on when to use one ELB over another in a particular situation.
- you might be presented with a network scenario where you need to determine when your ELB should be placed. Should it be an internal or external ELB? And we'll be using to serve encrypted traffic. In which case, what do you need to configure?
- if using HTTPS you'll need a service certificate perhaps issued by AWS Certificate Manager.

- assesses your ability of understanding how your ELBs react to targets in your target group that are marked as unhealthy following a health check. Now, does the ELB restart the instance? Does it launch another instance or does it just ignore it? Well for the ELB, it just ignores it and continues to send request to healthy instances. It's the job of auto-scaling to launch replacement instances not the ELB.
- AWS Lambda. Now this service isn't covered extensively on the exam but you certainly need to be aware of it and when it would be used.
- it is a serverless compute service designed to run in event-based environments to run application code without having to manage and provision your own EC2 instances. It's really cost-effective as you only pay for compute power when Lambda functions are invoked.
- In addition to being charged based on the number of times your function runs, known as invocations. So you might be presented with a question where you have an application that allows you to share photos that are uploaded to S3. But every time a new object is created, you want to process code to create a thumbnail of that object. What service would you use to do this with the least administrative effort? Now, this is a perfect example of when Lambda would be used. As its code-triggered by an event. And in this case, when a new object is uploaded is that event. And there were no resources to provision, to administer as it's serverless.
- The EC2 service can be broken down into the following components.
    - Amazon machine images, AMIs,
    - instant types,
    - instance purchasing options,
    - tenancy,
    - user data,
    - storage options and
    - security.
- Services:
    - **Elastic Compute Cloud, or EC2;** - It allows you to deploy virtual servers within your AWS environment and most people will require an EC2 instance within their environment as a part of at least one of their solutions.
    - **The Elastic Container Service, also known as ECS**- This service allows you to run Docker-enabled applications packaged as containers across a cluster of EC2 instances without requiring you to manage a complex and administratively heavy cluster management system.
    - **The Elastic Container Registry, or ECR;** - it provides a secure location to store and manage your docker images that can be distributed and deployed across your applications.
    - **The Elastic Container Service for Kubernetes, known as EKS;** - with EKS, AWS provides a managed service allowing you to run Kubernetes across your AWS infrastructure without having to take care of provisioning and running the Kubernetes management infrastructure in what's referred to as the control plane.
    - **AWS Elastic Beanstalk; -** AWS Elastic Beanstalk is an AWS-managed service that allows you to upload the code of your web application, along with the environment configurations, which will then allow Elastic Beanstalk to automatically provision and deploy the appropriate and necessary resources required within AWS to make the web application operational.
    - **AWS Lambda;** and
    - **AWS Batch:** All provisioning, monitoring, maintenance and management of the clusters themselves is taken care of by AWS, meaning there is no software to be installed by yourself.
    - **EC2 Auto Scaling. -**Auto Scaling is a mechanism that automatically allows you to increase or decrease your EC2 resources to meet the demand based off of custom defined metrics and thresholds.
        - Manual Scaling
        - Dynamic Scaling
        - Predictive Scaling
        - Scheduled Scaling
    - **AWS Elastic Load Balancer service (ELBs) -** is to help manage and control the flow of inbound requests destined to a group of targets by distributing these requests evenly across the targeted resource group.
        - Application Load Balancer
        - Network Load Balancer
        - Classic Load Balancer
    - **SSL or Secure Sockets Layer,** to give it its full name, is a cryptographic protocol, much like **TLS, Transport Layer Security**. Both SSL and TLS are used interchangeably when discussing certificates for your Application Load Balancer. The server certificates used by the ALB is an X.509 certificate, which is a digital ID that has been provisioned by a Certificate Authority and this Certificate Authority could be the **AWS Certificate Manager service** also known as **ACM**. This certificate is simply used to terminate the encrypted connection received from the remote client, and as a part of this termination process the request is then decrypted and forwarded to the resources in the ELB target group.
    - **The Gateway Load Balancer :** The AWS Gateway Load Balancer sends inbound and outbound traffic transparently over the same consistent route and using the same target. This implements sticky, transparent, and symmetric flow.

AWS Certified Solutions Architect - Associate exam.

- **Design Secure Architectures**
- **Design Resilient Architectures**
- **Design High-Performing Architectures**

• **Design Cost-Optimized Architectures**

Components that make up the AWS Global Infrastructure.
• **Availability Zones (AZs)**
• **Regions**
• **Edge Locations -** Edge Locations are primarily used by end users who are accessing and using your services.
• **Regional Edge Caches** - when data is requested at the Edge Location that is no longer available, the Edge Location can retrieve the cached data from the Regional Edge Cache instead of the Origin servers, which would have a higher latency.

>>
Lab - Create your first Amazon EC2 instance using the AWS Management Console.
>>

Amazon Elastic Compute Cloud (EC2) :
• EC2 allows you to launch different types of cloud instances and pay for them with a pay-per-use model.
• EC2 allows you to have operating system level control of your computing resources while running in Amazon's computing environment.
• EC2 reduces the time required to obtain and boot new server instances from days or weeks to minutes. This allows you to quickly scale capacity, both up and down, as your computing requirements change.
• EC2 allows you to build and configure your instances as you like, from your desired operating system to your applications.

Key Take aways :
• Understand the Instance States and other critical instance information
• Generate and use a Secure Shell (SSH) public/private key pair - understanding of SSH client software, protocol, and keys
• Connect to a running Linux instance using an SSH client
• Extract metadata about your running instance
• Terminate an instance

Command:
1. **get the key from keypair while launching an instance**
/Users/nanditha/Documents/Github/AWS/AWS_Solution_Lab: Key avaialble

2. connect either via the instance or through command line:

C02YR4H9LVCF:AWS_Solution_Lab nanditha$ **ssh -i /Users/nanditha/Documents/Github/AWS/AWS_Solution_Lab/keypair.pem ec2-user@54.189.186.33**
The authenticity of host '54.189.186.33 (54.189.186.33)' can't be established.
ED25519 key fingerprint is SHA256:xsVuhswrw19Obp/fCD+au3RIFxR41IUxeAw1rWgqje8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.189.186.33' (ED25519) to the list of known hosts.
@@@
@ WARNING: UNPROTECTED PRIVATE KEY FILE! @
@@@
Permissions 0777 for '/Users/nanditha/Documents/Github/AWS/AWS_Solution_Lab/keypair.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/Users/nanditha/Documents/Github/AWS/AWS_Solution_Lab/keypair.pem": bad permissions
ec2-user@54.189.186.33: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
C02YR4H9LVCF:AWS_Solution_Lab nanditha$ chmod 400 *
C02YR4H9LVCF:AWS_Solution_Lab nanditha$ ssh -i /Users/nanditha/Documents/Github/AWS/AWS_Solution_Lab/keypair.pem ec2-user@54.189.186.33

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|
```

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-31-206 ~]$ ls
[ec2-user@ip-172-31-31-206 ~]$ pwd
/home/ec2-user

-> Note: The IP address used below (169.254.169.254) is a special use address to return metadata information tied to EC2

instances.
->

3. **List all instance metadata by issuing the following command:**

[ec2-user@ip-172-31-31-206 ~]$ **curl -w "\n" http://169.254.169.254/latest/meta-data/**
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
events/
hostname
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
mac
managed-ssh-keys/
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/
system

NOTE: You can easily check the list of security groups attached to the instance, its ID, the hostname, or the AMI ID. The "-w" command-line option tells curl to write the output to standard output (STDOUT).

[ec2-user@ip-172-31-31-206 ~]$ **curl -w "\n" http://169.254.169.254/latest/meta-data/security-groups**
launch-wizard-1
[ec2-user@ip-172-31-31-206 ~]$ **curl -w "\n" http://169.254.169.254/latest/meta-data/ami-id**
ami-07f3ef11ec14a1ea3
[ec2-user@ip-172-31-31-206 ~]$ **curl -w "\n" http://169.254.169.254/latest/meta-data/hostname**
ip-172-31-31-206.us-west-2.compute.internal
[ec2-user@ip-172-31-31-206 ~]$ **curl -w "\n" http://169.254.169.254/latest/meta-data/instance-id**
i-001901bbf8407f373
[ec2-user@ip-172-31-31-206 ~]$ **curl -w "\n" http://169.254.169.254/latest/meta-data/instance-type**
t2.micro

4. Enter the following command to get the public SSH key of the attached key pair using the public-keys metadata:

[ec2-user@ip-172-31-31-206 ~]$ curl -w "\n" http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCKfLuFllzCfIljz2IKXx5lPDngCVpMa8ngJssKvoirWH+8A+zgQ+hbEff7htK9vBIzAKGe
q7mmPfzeJXzBSDjEgKc0a8Es2YlbGfFTxdizl/ImN85HQ1b1ACw8RcMGaqNslr5T8EbZ1syhd+wVQ7GBav4E+FZ1h/
VwPrIgGmXb6xeNltcTcRykprVWiHR04qNuy0QZ37f4PaOh9ExQ+oBhlKeZoLWJa2hXnTHoIzfudhLaHkmnwvDYPHvjSeYVMZL
VKxvOCiLBt7LcnUELq/SAnAIvVVcN0tG8oU/xUgLeHz0LC/iaRSo7YMh/y+9pvDc+Hlga6HFstI0YttsnWOX5 keypair

5. how to stop or terminate an instance from the AWS console

NOTE: When you are sure that you no longer need an instance, you can terminate it.
-> The specific instance and the data on the root volume (system disk) is not recoverable (by default) however. So be sure you don't need it before terminating an instance.
-> If you stop an instance, you can start it again later (and access data on all the disks).

[ec2-user@ip-172-31-31-206 ~]$
[ec2-user@ip-172-31-31-206 ~]$ Connection to 54.189.186.33 closed by remote host.
Connection to 54.189.186.33 closed.

>>>
Lab - Basics of Auto Scaling in Amazon Web Services - Working with Amazon EC2 Auto Scaling Groups and Network Load Balancer:
>>>

1. Scaling is the ability to increase or decrease the compute capacity of your application either by changing the number of servers (horizontal scaling) or by changing the size of the servers (vertical scaling).
2. Auto Scaling is well suited to applications that have stable demand patterns, or that experience hourly, daily, or weekly variability in usage.

3. **Groups:** Your EC2 instances are organized into groups so that they can be treated as a logical unit for the purposes of scaling and management. When you create a group, you can specify its minimum, maximum, and desired number of EC2 instances.

4. **Launch configurations:** Your group uses a launch configuration as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.

5. **Launch template:** A launch template is similar to a launch configuration, in that it specifies instance configuration information. ... However, defining a launch template instead of a launch configuration allows you to have multiple versions of a template. With versioning, you can create a subset of the full set of parameters and then reuse it to create other templates or template versions.

6. Auto Scaling groups are commonly paired with load balancers that evenly distribute requests across all the instances in the group.

7. **Elastic Load Balancing (ELB)** automatically distributes incoming application traffic across multiple Amazon EC2 instances. They enable you to achieve greater fault tolerance in your applications and seamlessly provide the correct amount of load balancing capacity needed in response to incoming application requests. ELB detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.

8. Create a network load balancer: you created a Network Load Balancer with a target group ready to service HTTP requests on port 80. This load balancer will be used as the front-end to a website. The website will run on EC2 instances that are created via an Auto Scaling group.

-> The target type option allows you to specify IP addresses or a Lambda function in addition to Instances. Using an IP address gives you the ability to use a network load balancer with compute instances outside of AWS
-> Note: Ensure you set the protocol to TCP. Because network load balancers operate at layer 4 and aren't HTTP aware, if you set the protocol as HTTP you will be unable to use the target group with your network load balancer.
-> You must enable Cross-zone load balancing to achieve the highest level of availability. Without enabling this feature, clients could cache the DNS address of the load balancer node in one availability zone and that node would only distribute requests to instances within the availability zone. Cross-Zone Load Balancing allows every load balancer node to distribute requests across all availability zones, although for the Network Load Balancer there are data transfer charges when this feature is enabled. (There are no data charges for other types of load balancers)
-> The deregistration delay specifies how long the load balancer should wait before removing an instance from the target group.

9. When you create the Launch Template you will include information such as the Amazon machine image ID (AMI) to use for launching the EC2 instance, the instance type, key pairs, security groups, and block device mappings, among other configuration settings. When you create your Auto Scaling group, you must associate it with a Launch Template..
-> The Launch Template essentially contains the blueprint or DNA for the exact type of instance that should be launched. Hence, when auto-scaling, each instance is guaranteed to be just like the last one. It's repeatable, scalable, and reliable.
-> Note: Enabling detailed monitoring incurs an extra cost.
-> Warning: The EC2 instances will never reach 100% CPU Utilization due to the limitations of the burstable credit. They should reach an usage of about 80%.

10. create an auto-scaling group using the launch template :
-> An Auto Scaling group starts by launching the minimum number (or the desired number, if specified) of EC2 instances and then increases or decreases the number of running EC2 instances automatically according to the conditions that you define.
-> Auto Scaling also maintains the current instance levels by conducting periodic health checks on all the instances within the Auto Scaling group.

-> If an EC2 instance within the Auto Scaling group becomes unhealthy, Auto Scaling terminates the unhealthy instance and launches a new one to replace the unhealthy instance. This automatic scaling and maintenance of the instance in an Auto Scaling group is the core value of the Auto Scaling service. It's what puts the "elastic" in EC2.
-> Auto-scaling groups allow you to scale out (add more instances) or scale in (remove instances) based on metrics such as:
- Average CPU utilization
- Network traffic (ingress and egress)
- Application Load Balancer request counts

-> In this lab step, you created an Auto Scaling group using a launch template. You defined a scaling policy to scale up or down based on the average CPU utilization of all the instances in the Auto Scaling group. The scaling policy makes use of CloudWatch metrics to trigger an alarm to cause a scale-in or scale-out event. You also configured the Auto Scaling group to automatically register its instances to a target group of a Network Load Balancer.

: command line to run stress causing the CPU utilization to increase for five minutes:
>>>
Lab - Create your first AWS Lambda
>>>
-> AWS Lambda is a service released for Developer Preview in April 2015.
-> you can write a stateless Lambda function that will be triggered to react to certain events (or HTTP endpoints)
-> When running a job processing server in EC2, you are charged for compute-time as long as your instance is running. Contrast that with Lambda, where you are only charged while actually processing a job, on a 1ms basis. Basically, you never pay for idle time.
-> Lambda a great fit for spiky or infrequent workloads because it scales automatically and minimizes costs during slow periods.
-> The event-based model Lambda provides makes it perfect for providing a backend for mobile clients, IoT devices, or adding no-stress asynchronous processing to an existing application, without worrying too much about scaling your compute power.
-> Elastic Beanstalk is a PaaS that lets you deploy code without worrying about the underlying infrastructure. However, compared to Lambda, it does provide more choices and controls. You can deploy complete applications to Elastic Beanstalk using a more traditional application model compared to deploying individual functions in Lambda.
-> Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Container Service for Kubernetes (Amazon EKS) are centered around containers compared to the individual functions of Lambda. ECS and EKS require less managerial overhead compared to running containers on EC2 instances, but generally require some operational expertise. Lambda is ideal for developers who just want to focus on their code.
-> Simple Workflow Service is a coordination service, and you must provision workers to complete your tasks.
-> AWS Lambda can receive these messages over Amazon Simple Notification Service (SNS), and you can hook as many functions as you like to the same SNS topic.

>>>
Exam 1+2 on below: Knowledge Check: Compute (SAA-C03)
>>>

Amazon Elastic Cloud Compute (EC2)
AWS Lambda
Elastic Beanstalk
Amazon Elastic Container Services (ECS)
Amazon Elastic Container Registry (ECR)
Amazon Elastic Kubernetes Service (EKS)

1. kubectl is a command line interface, for running commands against Kubernetes clusters. The AWS IAM authenticator is a tool that allows you to use AWS IAM credentials to authenticate against Kubernetes clusters. And eksctl provides a nice abstraction for creating clusters.
2. An AWS Elastic Beanstalk environment is a collection of AWS resources running an application version. You can deploy multiple environments when you need to run multiple versions of an application. For example, you might have development, integration, and production environments. An application version is a very specific reference to a section of deployable code. The application version, not the environment, will point typically to S3, where the deployable code may reside. An environment is the application version deployed on AWS resources and not just an EC2 instance with your uploaded code. The platform is a culmination of components in which you can build your application upon using Elastic Beanstalk.
3. The control plane schedules containers onto nodes. The term scheduling does not refer to time in this context. Scheduling, in this case, refers to the decision process of placing containers onto nodes in accordance with their declared, compute requirements. The Control Plane also tracks the state of all Kubernetes objects by continually monitoring the objects. So in EKS, AWS is responsible for provisioning, scaling and managing the control plane and they do this by utilizing multiple availability zones for additional resilience. However, Provision and maintain worker nodes is not carried out
4. Spot Instances utilizes spare computing capacity for up to 90% savings over On-Demand instances. There are four

general categories of time-flexible and interruption-tolerant tasks that work well with Spot Instances: delayable tasks, optional tasks, tasks that can be sped up by adding additional computing power, and tasks that require a large number of compute instances that you can't access any other way.

5. On-Demand instances are cost-effective if you want to run short-term, unpredictable workloads that cannot be interrupted.
6. Reserved Instances must be purchased for a minimum of 1-3 year terms and are best for long-term, predictable workloads.
7. Savings plans offer flexible pricing on EC2 and Fargate usage but in exchange for a commitment to a consistent amount of usage.
8. ECS can manage all EC2 instances for all ECS clusters but also allows you to manage instances directly through the ECS service.
9. Amazon ECS allows you two deployment options, a Fargate launch and an EC2 launch. In a Fargate launch, the user defines the service requirements, and the service manages the rest. In an EC2 launch, the user is responsible for patching and scaling instances, and must specify the instance type and number of containers in a cluster.
10. Elastic Container Registry is a secure location to store and manage Docker images. It is a fully managed service. Elastic Container Service can pull images from registries within ECR and deploy them within ECS clusters.
11. A trigger is essentially an operation from an event source that causes the function to invoke, and so essentially is triggering that function.
12. You want to create an application that only runs when your shop is open from 9 am to 5 pm Monday through Thursday. On-demand Capacity Reservations payment option would offer the greatest cost savings
13. On-Demand Capacity Reservations enable you to reserve compute capacity for your Amazon EC2 instances in a specific Availability Zone for any duration. This gives you the ability to create and manage Capacity Reservations independently from the billing discounts offered by Savings Plans or Regional Reserved Instances
14. By creating Capacity Reservations, you ensure that you always have access to EC2 capacity when you need it, for as long as you need it. You can create Capacity Reservations at any time, without entering into a one-year or three-year term commitment. The capacity becomes available and billing starts as soon as the Capacity Reservation is provisioned in your account. When you no longer need it, cancel the Capacity Reservation to release the capacity and to stop incurring charges.
15. Using the Lambda qualifiers section allows you to change between different versions of your function.
16. ECR registries contain multiple repositories, which contain multiple Docker images.
17. Amazon Elastic Container Service allows you to define tasks through a declarative JSON template called a Task Definition. Within a Task Definition you can specify one or more containers required for your task, including the Docker repository and image, memory and CPU requirements, shared data volumes, and how the containers are linked to each other. You can launch as many tasks as you want from a single Task Definition file that you can register with the service. Task Definition files also allow you to version control your application specification.
18. Considering Amazon EC2, dedicated host instances purchase option below allows customers to select and control a specific, physical server within an AWS data center solely for their use
19. When you launch instances on a Dedicated Host, the instances run on a physical server that is dedicated for your use. While Dedicated instances also run on dedicated hardware, Dedicated Hosts provide further visibility and control by allowing you to place your instances on a specific, physical server. This enables you to deploy instances using configurations that help address corporate compliance and regulatory requirements.
20. If you have to host a server with a specific region and availability zone for a year, the cheapest way to start that instance is to use the Reserved Instance
21. Amazon EC2 Reserved Instances can be a powerful and cost-saving strategy for running your business. Amazon EC2 Reserved Instances allow you to reserve Amazon EC2 computing capacity for 1 or 3 years, in exchange for a significant discount (up to 75%) compared to On-Demand instance pricing. Reserved Instances can significantly lower your computing costs for your workloads and provide a capacity reservation so that you can have confidence in your ability to launch the number of instances you have reserved when you need them.
22. A video production company currently uses “local transcoding” and relies on video editing and compression software to transcode files on a user’s computer. They have encountered latency in the editing process due to hardware limitations. Compute-optimized instance types would be recommended to solve the problem of processing latency
23. Compute-optimized instances would be the best solution for avoiding processing latency because this family of instances has the highest performing processors installed and are used in high-performance applications such as video encoding. While Storage-optimized instances offer low latency because they have SSD-backed instant storage and very high I/O, input/output performance, including very high input/output operations per second, they are best used for high capacity analytic workloads, such as data file systems and log processing applications. General-purpose, instance types have a balanced mix of CPU memory and storage making them ideal for small to medium databases, tests, development servers, and back-end servers. Memory-optimized, instance types are primarily used for large-scale enterprise-class in-memory applications, such as performing real-time processing of unstructured data, for example, applications such as Microsoft Sharepoint.
24. Regarding AWS Elastic Beanstalk, An environment configuration identifies a collection of parameters and settings that define how an environment and its associated resources behave.
25. In AWS Elastic Beanstalk, an environment configuration identifies a collection of parameters and settings that define how an environment and its associated resources behave. When you update an environment’s configuration settings, AWS Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources (depending on the type of change).

26. The Throttle option is closely linked to the concurrency setting that we just talked about. Selecting this option sets the reserve concurrency limit of your function to zero.
27. You are managing a VPC for a client in the marketing industry. They have several tasks, such as flyer and coupon creation, which are not mission-critical and can just be restarted if the application terminates. You recommend using spot instances for the greatest cost savings. Under below circumstances will a spot instance terminate:
    A. The customer terminates them.
    B. The spot price is higher than the user's maximum price
    C. There is not enough unused capacity to meet the demand for spot instances.
28. Spot instances can be terminated by the customer if there is not enough unused capacity to meet the demand for spot instances. Spot instances are billed hourly, but you will not be charged for the hour if the instance is terminated by AWS. If you terminate the instance yourself, you will be charged for the full hour in which the termination occurred.
29. For poll-based event sources, the invocation type is always synchronous.
30. Memory-optimized Amazon EC2 instance family is ideal for applications that manage real-time unstructured data processing, or distributed web cache stores
31. Memory-optimized instance types are primarily used for large-scale, enterprise-class, in-memory applications, such as performing real-time processing of unstructured data or for in-memory databases such as SAP HANA.
32. AWS Fargate AWS service enables AWS users to run containers on Amazon ECS without provisioning and managing the host EC2 instances.
33. When using the Fargate launch type with tasks within your cluster, Amazon ECS manages your cluster resources.
34. The worker environment applies mainly to backend processes, while the webserver environment is associated with HTTP requests.
35. When installing tools for operations in Amazon EKS, The kubectl tool is a command-line interface, for running commands against Kubernetes clusters.
36. Amazon EC2 provides Amazon Machine Images (AMIs), which are preconfigured templates for your instances.
37. Storage-optimized Amazon EC2 instance family offers SSD-backed instance storage for high I/O performance
38. Storage optimized; as expected, these are optimized for enhanced storage. Instances in this family use SSD-backed instance storage for low latency and very high input/output performance, including very high IOPS, which stands for input-output operations per second. These are great for analytic workloads and NoSQL databases, data file systems and lock processing applications.
39. Amazon ECR repository policies are a subset of IAM policies that are scoped for, and specifically used for, controlling access to individual Amazon ECR repositories. IAM policies are generally used to apply permissions for the entire Amazon ECR service but can also be used to control access to specific resources as well.
40. Both Amazon ECR repository policies and IAM policies are used when determining which actions a specific user or role may perform on a repository. If a user or role is allowed to perform an action through a repository policy but is denied permission through an IAM policy (or vice versa) then the action will be denied.
41. instance families can be summarized as follows.
    A. **Micro instances,** these instances have a low cost against them due to the minimal amount of CPU and memory power that they offer. These are ideal for very low throughput use cases such as low traffic websites.
    B. **General-purpose, i**nstance types within this family have a balanced mix of CPU memory and storage making them ideal for small to medium databases, tests and development servers and back-end servers.
    C. **Compute optimized,** as the name implies instance types within this family have a greater focus on compute. They have the highest performing processes installed allowing them to be used for high-performance front end servers, web servers, high-performance science and engineering applications and video encoding and batch processing.
    D. **GPU,** GPU stands for graphics processing unit. And so the instances within this family are optimized for graphic intensive applications. FPGA, this family of instances allows you to customize field programmable gate arrays. To create application specific hardware accelerations when used with applications that use massively parallel processing power such as genomics and financial computing.
    E. **Memory optimized,** this family include instance types that are primarily used for large-scale enterprise class in-memory applications, such as performing real time processing of unstructured data. They are also ideal for enterprise applications such as Microsoft SharePoint. These instances of the lowest cost per gigabyte of RAM against all other instance families.
    F. **Storage optimized,** as expected these are optimized for enhanced storage. Instances in this family use SSD backed instant storage for low latency and very high I/O, input/output performance, including very high IOPS which is input/ output operations per second. And these are great for analytic workloads and no SQL databases. Data file systems and log processing applications.
42. The different EC2 payment options are as follows,
    A. **on-demand instances:** These are EC2 instances that you can launch at any time and have it provisioned and available to you within minutes. You can use this instance for a shorter time or for as long as you need before terminating the instance. These instances have a flat rate and is determined on the instance type selected and is paid by the second. On-demand instances are typically used for short term uses where workloads can be irregular and where workload can be interrupted. Many users of AWS use on-demand instances within their testing and development environments. And when you stop or terminate your on-demand instance you'll stop paying for the compute resource.
    B. **reserved instances:** Reserved instances allow you to purchase a discount for an instance type with set criteria for a set period of time in return for a reduced cost compared to on-demand instances. This reduction can be as much as

75%. These reservations against instances must be purchased in either one or three year time frames. Further reductions can be achieved with reserved instances depending on which payment methods you select. There are three options available to you:
    a. firstly all upfront. The complete payment for the one or three year reservation is paid. And this offers the largest discount and no further payment is required regardless of the number of hours the instance is used.
    b. Partial upfront, here a smaller upfront payment is made and then a discount is applied to all remaining hourse during the term.
    c. And finally no upfront, no upfront or partial payments are made and the smallest discount of the three models is applied to all remaining hours in the term.

C. Reserved instances are used for long-term predictable workloads allowing you to make full use of the cost savings to be had when using compute resources offered by EC2.
D. **scheduled instances:** Scheduled instances, these are similar to reserved instances and the fact that you pay for the reservations of an instance on a recurring schedule, either daily, weekly or monthly. For example you might have a weekly task that is scheduled that performs some kind of bulk processing for a number of hours at the same time every week. With scheduled instances you could set up a scheduled instance to run during that set timeframe once a week. And this prevents you for having to use the on-demand instances which would incur a higher price. You should note that when using scheduled instances but even if you didn't use the instance you would still be charged. This allows you to provision instances for scheduled workloads that are not continuously running. Which is where a reserved instance would be the preferred choice.
E. **spot instances :** Spot instances allows you to bid for unused EC2 compute resources, however your resource is not guaranteed for a fixed period of time. To you to spot instance you must bid higher than the current spot price which is set by AWS. And this spot price fluctuates depending on supply and demand of the unused resource. If your bid price for an instance type is higher than the spot price, then you'll purchase that instance. But as soon as your bid price becomes lower than the fluctuating spot price, you will be issued a two-minute warning before the instance automatically terminates and is removed from your AWS environment. The bonus for spot instances is that you can bid for large EC2 instances at a very low cost point saving a huge amount on cost. Due to the nature of how the instances can be suddenly removed from your environment, spot instances are only useful for processing data and applications that can be suddenly interrupted. Such as batch jobs and background processing of data.
F. **on-demand capacity reservations:** Capacity reservations allows you to reserve capacity for your EC2 instances based on different attributes. Such as instance type, platform and tenancy et cetera. Within a particular availability zone for any period of time. This ensures that you always have the available number of instances you require within a specific availability zone immediately. This capacity reservation could also be used in conjunction with your reserved instances discount providing you additional savings.

43. EC2 tenancy and this relates to what underlying host your EC2 instance will reside on. So essentially the physical server within an AWS data center.
    A. **Shared tenancy,** this option will launch your EC2 instance on any available host with the specified resources required for your selected instance type. Regardless of which other customers and users also have EC2 instances running on the same host, hence the share tenancy name. AWS implement advanced security mechanisms to prevent one EC2 instance from accessing another on the same host. How the security is applied and operated is maintained by AWS themselves.
    B. **Dedicated tenancy,** this includes both dedicated instances and dedicated hosts. Dedicated instances are hosted on hardware that no other customer can access. It can only be accessed by your own AWS account. You may be required to launch your instances as a dedicated instance due to internal security policies or external compliance controls. Dedicated instances do incur additional charges due to the fact you are preventing other customers from running EC2 instances on the same hardware and so there will likely be unused capacity remaining. However the hardware might be shared by other resources you have running in your own account.
    C. **Dedicated host,** a dedicated host is effectively the same as dedicated instances. However they offer additional visibility and control, how you can place your instances on the physical host. They also allow you to use your existing licenses, such as PA-VM license or Windows Server licenses et cetera. Using dedicated hosts give you the ability to use the same host for a number of instances that you want to launch and align with any compliance and regulatory requirements. If you don't need to address any compliance or security issues that require dedicated tenancy, then I recommend using shared tenancy to reduce your overall costs.
44. Storage for EC2 can be classified between two distinct categories,
    A. **Persistent storage** is available by attaching **elastic block storage EBS volumes**. EBS volumes are separate devices from the EC2 instance itself. And so it's not physically attached like ephemeral storage is. EBS volumes are considered network attached storage devices which are then logically attached to the EC2 instance via the AWS network. This principle is not dissimilar to attaching an external hard disk to your home laptop or PC. With the external hard disk represent your EBS volume and your PC represents your EC2 instance. The data on EBS volumes are automatically replicated to other EBS volumes within the same availability zone for resiliency which is managed by AWS. You can disconnect an EBS volume from your EC2 instance and the data will remain intact. Allowing you to reattach it to another EC2 instance if required. You can also implement encryption on these volumes if needed and take backup snapshots of all the data on the volume to S3. EBS volumes can be created in different sizes again with different performance capabilities depending on your requirements.
    B. **Ephemeral storage** is created by some EC2 instances themselves using a local storage on the underlying host known as instance back storage. Ephemeral storage or instance backed storage is the storage that is physically

attached to the underlying host on which the EC2 instance resides on. Looking back at our previous example, this would be similar to your own laptop or PC's hard disk. There is a difference here though, with AWS EC2 instances as soon as the instance is stopped or terminated all saved data on a ephemeral storage is lost. If you reboot your instance then the data will remain but not if you stop it. Therefore if you have data that you need to retain it is not recommended that you use instance backed storage for this data. Instead use EBS volumes for persistent data storage. Unlike EBS volumes you are unable to detach ephemeral instance store volumes from the instance.

45. A security group is essentially an instance level firewall allowing you to restrict both ingress and egress traffic by specifying what traffic allowed to communicate with it. You can restrict this communication by source ports and protocols for both inbound and outbound communication. Your instances are then associated with this security group.
46. A key pair, as the name implies, is made up of two components, a public key and a private key. The function of key pairs is to encrypt the login information for Linux and Windows EC2 instances. And then decrypt the same information allowing you to authenticate onto the instance. The public key encrypts data such as the username and password. For Windows instances, the private key is used to decrypt this data allowing you to gain access to the login credentials including the password. For Linux instances the private key is used to remotely connect onto the instance via SSH. The public key is held and kept by AWS, and the private key is your responsibility to keep and ensure that it is not lost or compromised.
47. Once you have authenticated to the EC2 instance the first time, you can set up additional less privileged access controls such as local windows accounts allowing other users to connect and authenticate to or even utilize Microsoft Active Directory. One final point regarding security on your EC2 instance it is your responsibility to maintain and install the latest OS and security patches released by the OS vendor as dictated within the AWS shared responsibility model.
48. There are two types of status checks.
    A. **System status checks:** If the system status check fails then it is likely to be an issue with the underlying host rather than a configuration issue with your EC2 instance. Common issues that trigger system status checks to fail are loss of power, loss of network connectivity and hardware and software issues on the underlying host. Basically a system status check failure is out of our control as the fault lies with components that AWS are responsible for. The best way to resolve this will be to stop the instance and restart. This is likely to cause the instance to launch on another physical host resolving the problem. Do not reboot the instance as this will cause the instance to continue running on the same physical server.
    B. **Instance status checks,** these differ from system status checks as if this fails then it would likely require your input to help them resolve the issue. This check looks at the EC2 instance itself, rather than focusing on the underlying hosts. Common issues that trigger these checks to fail are incorrect network configuration, corrupted file systems, exhausted memory or incompatible kernel. These faults will require you to troubleshoot and resolve the issue, for example changing the network configuration.
49. **Amazon EC2 Container Service,** commonly known as **Amazon ECS.** This service allows you to run Docker-enabled applications packaged as containers across a cluster of EC2 instances without requiring you to manage a complex and administratively heavy cluster management system. The burden of managing your own cluster management system is abstracted with the Amazon ECS service by passing that responsibility over to AWS, specifically though the use of AWS Fargate.
    A. **AWS Fargate** is an engine used to enable ECS to run containers without having to manage and provision instances and clusters for containers.
    B. **Docker** is piece of software that allows you to automate the installation and distribution of applications inside Linux Containers.
    C. A **Container** holds everything that an application requires to enable it to run from within it's isolated container package. This may include system libraries, code, system tools, run time, etcetera. But it does not include an operating system like a virtual machine does, and so reduces overhead of the actual container itself. Containers are decoupled from the underlying operating system, making Container applications very portable, lightweight, flexible, and scalable across a cloud environment. This ensures that the application will always run as expected regardless of it's deployment location.
50. When launching your ECS cluster you have the option of two different deployment models: a Fargate launch and an EC2 launch.
    A. **The Fargate launch** requires far less configuration and simply requires you to specify the CPU and memory required, define the networking and IAM policies in addition to you having to package your applications into containers.
    B. **an EC2 launch** you have a far greater scope of customization and configurable parameters. For example, you are responsible for patching and scaling your instances, and you can specify which instance types you used, and how many containers should be in a cluster.
51. **Elastic Container Registry service, known as ECR**: the EC2 Container Service, as it provides a secure location to store and manage your docker images that can be distributed and deployed across your applications. This is a fully managed service, and as a result, you do not need to provision any infrastructure to allow you to create this registry of docker images. This is all provisioned and managed by AWS. This service is primarily used by developers, allowing them to push, pull, and manage their library of docker images in a central and secure location.
    A. **The ECR registry** is the object that allows you to host and store your docker images in, as well as create image repositories. Within your AWS account, you will be provided with a default registry. When your registry is created, then by default, the URL for the registry is as follows: https://aws_account_id.dkr.ecr.region.amazonaws.com where you'll need to replace the red text with your own information that is applicable to your account or medium. Your account will have both read and write access by default to any images you create within the registry and any repositories. Access to your registry and images can be controlled via IAM policies in addition to repository policies as well, to enforce

tighter and stricter security controls. As the docker command line interface doesn't support the different AWS authentication methods that are used, then before your docker client can access your registry, It needs to be authenticated as an AWS user, which will then allow your client to both push and pull images. And this is done by using an authorization token. To begin the authorization process to allow your docker client to communicate with the default registry, you can run the get login command using the AWS CLI, as shown: aws ecr get-login --region region --no-include-email where the red text should be replaced with your own region. This will then produce an output response, which will be a docker login command docker login -u AWS -p password https://aws_account_id.dkr.ecr.region.amazonaws.com You must then copy this command and paste it into your docker terminal which will then authenticate your client and associate a docker CLI to your default registry. This process produces an authorization token that can be used within the registry for 12 hours, at which point, you will need to re-authenticate by following the same process.

52. Using policies from both IAM and repository policies, you can assign permissions to each repository allowing specific users to perform certain actions, such as performing a push or pull IP line. you can control access to your repository and images using both IAM policies and repository policies. There are a number of different IAM managed policies to help you control access to ECR, these being the three shown on the screen.
    A. AmazonEC2ContainerRegistryFullAccess
    B. AmazonEC2ContainerRegistryPowerUser
    C. AmazonEC2ContainerRegistryReadOnly
53. Once you have configured your registry, repositories, and security controls, and authenticated your docker client with ECR, you can then begin storing your docker images in the required repositories, ready to then pull down again as and when required.
54. **Amazon Elastic Container Service for Kubernetes, known as EKS:** with EKS, AWS provides a managed service allowing you to run Kubernetes across your AWS infrastructure without having to take care of provisioning and running the Kubernetes management infrastructure in what's referred to as the control plane. You, the AWS account owner, only need to provision and maintain the worker nodes.
    A. **Kubernetes Control Plane:** There are a number of different components that make up the control plane and these include a number of different APIs, the kubelet processes and the Kubernetes Master, and these dictate how kubernetes and your clusters communicate with each other. The control plane itself is run across master nodes. The control plane schedules containers onto nodes. The term scheduling does not refer to time in this context. Scheduling, in this case, refers to the decision process of placing containers onto nodes in accordance with their declared, compute requirements. The Control Plane also tracks the state of all kubernetes objects by continually monitoring the objects. So in EKS, AWS is responsible for provisioning, scaling and managing the control plane and they do this by utilising multiple availability zones for additional resilience.
    B. **Worker nodes:** Kubernetes clusters are composed of nodes and the term cluster refers to the aggregate of all of the nodes. A node is a worker machine in Kubernetes and runs as an on-demand EC2 instance and includes software to run containers managed by the Kubernetes control plane. For each node created, a specific AMI is used which also ensures docker and kubelet in addition to the AWS IAM authenticator is installed for security controls. These nodes are what us as the customer are responsible for managing within EKS. Once the worker nodes are provisioned they can then connect to EKS using an endpoint.
55. Before you begin working with EKS you need to configure and create an IAM service-role that allows EKS to provision and configure specific resources. This role only needs to be created once and can be used for all other EKS clusters created going forward. The role needs to have the following permissions policies attached to the role:
    A. AmazonEKSServicePolicy and
    B. AmazonEKSClusterPolicy
56. **AWS Elastic Beanstalk** is an AWS managed service that allows you to upload the code of your web application, along with the environment configurations, which will then allow Elastic Beanstalk to automatically provision and deploy the appropriate and necessary resources required within AWS to make the web application operational.
57. With enhanced health, the Elastic Beanstalk health agent reports metrics to Elastic Beanstalk every **10sec.** It then uses the metrics provided by the health agent to determine the health status of each instance in the environment..
58. There is no cost associated with Elastic Beanstalk, however, any resources that are created on your application's behalf, such as EC2 instances, you will be charged for as per the standard pricing policies at the time of deployment.
    A. **The application version.** An application version is a very specific reference to a section of deployable code. The application version will point typically to S3, simple storage service to where the deployable code may reside. In Elastic Beanstalk, an application version refers to a specific, labeled iteration of deployable code for a web application.
    B. **The environment.** An environment refers to an application version that has been deployed on AWS resources. These resources are configured and provisioned by AWS Elastic Beanstalk. At this stage, the application is deployed as a solution and becomes operational within your environment. The environment is comprised of all the resources created by Elastic Beanstalk and not just an EC2 instance with your uploaded code. An environment is a collection of AWS resources running an application version. **Each environment runs only one application version at a time,** however, **you can run the same application version or different application versions in many environments simultaneously.** When you create an environment, Elastic Beanstalk provisions the resources needed to run the application version you specified.
    C. **Environment configurations.** An environment configuration is a collection of parameters and settings that dictate how an environment will have its resources provisioned by Elastic Beanstalk and how these resources will behave. In

AWS Elastic Beanstalk, an environment configuration identifies a collection of parameters and settings that define how an environment and its associated resources behave. When you update an environment's configuration settings, AWS Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources (depending on the type of change).

D. **The environment tier.** This component reflects on how Elastic Beanstalk provisions resources based on what the application is designed to do. If the application manages and handles HTTP requests, then the app will be run in a web server environment. If the application does not process HTTP requests, and instead perhaps pulls data from an SQS queue, then it would run in a worker environment. I shall cover more on the differences between the web server and work environment shortly.
    a. **The web server tier.** The web server environment is typically used for standard web applications that operate and serve requests over HTTP port 80. This tier will typically use the following AWS resources in the environment. Route 53. When an environment is created by Elastic Beanstalk, it has an associated URL. Using the CNAME record in Route 53, this URL is aliased to an Elastic Load Balancer, an ELB. If you need more information on Route 53, please see our existing course here.
        1. **Elastic Load Balancer.** For every environment, you should have at least one ELB sitting in front of your EC2 instances that is referenced by Route 53 as just explained. This ELB will also integrate with an autoscaling group.
        2. **Auto scaling.** Again, for every environment, you will also have an auto scaling group that will manage the capacity planning of your applications based on the load received. As and when required it will both add and remove EC2 instances to ensure your application meets the demands of its users. In AWS Elastic Beanstalk, 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 **AWS Elastic Beanstalk supports only one Auto Scaling group per environment.**
        3. **EC2 instances.** Within the environment, AWS Elastic Beanstalk will create a minimum of one EC2 instance to run your application. This EC2 instance will be a part of the auto scaling group.
        4. **Security groups.** Your EC2 instances will be governed by a security group, which by default will allow port 80 open to everyone. An important component within the environment is actually installed on every EC2 instance provisioned. This component is the Host Manager.
        5. **The Host Manager** has a number of different key and fundamental responsibilities. And these include to aid in the deployment of your application, it collates different metrics and different events from the EC2 instance which can then be reviewed from within the console or via the AWS CLI or API, it generates instance-level events, it monitors both the application log files and the application server itself. It will also patch instance components, and finally it will manage the log files, allowing them to be published at S3.
    b. **worker tier:** worker environment is slightly different and are used by applications that will have a backend processing task, which will interact with AWS SQS, the Simple Queue Service. This tier typically uses the following AWS resources within the environment,
        1. **an SQS Queue.** If you don't already have an SQS Queue operational and configured, then as a part of the creation of the worker environment AWS Elastic Beanstalk will create one for you.
        2. **An IAM Service Role.** To allow your EC2 instances to monitor queue activity in the SQS Queue, each EC2 instance will have an associated instance profile role which contains a section within the policy.
        3. **Auto scaling.** An auto scaling group is created to ensure that performance isn't impacted based on load.
        4. **EC2 instances,** a minimum of one EC2 instance is used and is attached to the auto scaling group, and each EC2 instance in the environment will read from the same SQS Queue.
        5. Whereas the web tier used the Host Manager to perform some key tasks for Elastic Beanstalk, instead within a worker tier a **Daemon** is installed on every EC2 instance to pull requests from your SQS Queue. It will also then send data to the application allowing it to process the message. This is why the instance profile role is required to ensure permissions are given to read from a queue.
    c. there are clear differences between the two tiers, and it's likely that you will use the two tiers in conjunction with each other, decoupled by the use of the simple queue service, allowing each environment to scale independently of one another depending on demand through auto scaling and Elastic Load Balancing.
    d. for additional customization over what is provisioned by the service itself within these environments, then you can develop and add your own **Elastic Beanstalk configuration files** within your application source code. These are either written in a YAML or JSON based format. And these customization files need to be saved within the **.config** file extension and then stored within the **.ebextensions folder** of your source code.

E. **The configuration template.** This is the template that provides the baseline for creating a new, unique environment configuration.

F. **Platform.** The platform is a culmination of components in which you can build your application upon using Elastic Beanstalk. These comprise of the operating system of the instance, the programming language, the server type, web or application, and components of Elastic Beanstalk itself, and as a whole can be defined as a platform.

G. **Applications.** Within Elastic Beanstalk, an application is a collection of different elements, such as environments, environment configurations, and application versions. In fact, you can have multiple application versions held within a single application. You can deploy your application across one of two different environment tiers, either the web server tier or the worker tier.
    a. **The web server environment** is typically used for standard web applications that operate and serve requests over HTTP port 80. This tier will typically use service and features such as Route 53, Elastic Load Balancing, Auto

Scaling, EC2, and Security Groups.

b. **The worker environment** is slightly different and are used by applications that will have a back-end processing task that will interact with AWS SQS, the Simple Queue Service. This tier typically uses the following AWS resources in this environment, an SQS Queue, an IAM Service Role, Auto Scaling, and EC2.

H. Deployment Options:

a. **All at once deployment** is the default choice if you don't specify any other option. If you needed to update your application within your Elastic Beanstalk environment. Using the all at once option will simply roll out the application to your resources all at the same time. And this would, of course, cause a disruption to your application while the update was in progress, which would, in turn, affect your end users.

b. **Rolling,** with a rolling deployment you are able to minimize the amount of disruption that is caused when using the all at once approach. When performing a rolling deployment, Elastic Beanstalk will deploy your application in batches, performing an update to just a portion of your resources at a time. Once the update is complete it will then perform the update on the next batch. This would mean that you have two different versions of the application running at the same time for a very short period. However, it also means that you can still serve requests and process information through your application while the deployment is gradually rolled out for your infrastructure.

c. **Rolling with additional batch.** The rolling with additional batch is much the same principle as rolling. Your environment is updated in batches until all your resources have the new update. However, with rolling deployments there is an impact to your available resources while the update is being applied. For example, let's say during a rolling deployment Elastic Beanstalk creates four different batches. While the update is being applied, your application availability takes a 25% hit while one batch is being updated. With rolling with additional batch, Elastic Beanstalk adds another batch of instances within your environment to your resource pool to ensure application availability is not impacted. So in this case, your deployment would have five batches. And while the update was being applied to one of your existing batches, you would still have four remaining batches of instances maintaining operation. On completion of the update to all batches, Elastic Beanstalk would then terminate this additional batch of instances.

d. **Immutable,** immutable deployments will create an entirely new set of instances in parallel to your existing and older resources. These new instances will be served through a temporary autoscaling group behind your elastic load balancer. This means for a short period of time your environment would essentially double in size. However, once your new instances are deployed and have passed all the health checks the old environment would be removed and the autoscaling group updated. If your health checks fail for the new instances then Elastic Beanstalk would terminate them and delete the autoscaling group and traffic would continue to be served to your original environment.

59. **AWS Lambda** is a serverless compute service which has been designed to allow you to run your application code without having to manage and provision your own EC2 instances. This saves you having to maintain and administer an additional layer of technical responsibility within your solution. Instead, that responsibility is passed over to AWS to manage for you.
60. Although it's named serverless, it does of course require service, or at least compute power, to carry out your code requests, but because the AWS user does not need to be concerned with what's managing this compute power, or where it's provisioned from, it's considered serverless from the user perspective.
61. AWS starts and stops Lambda compute resources as needed.
62. The Lambda service is highly scalable because AWS will automatically start and stop Lambda compute resources as needed according to demand. Paying for the code only while it runs in Lambda is an example of how it optimizes costs per compute function. Specifying how many function instances are used and adding unlimited logging statements are not primary examples of Lambda's scalability because AWS manages the function instances, while adding logging statements is optional within Lambda functions.
63. AWS Lambda charges compute power per 100 milliseconds of use only when your code is running, in addition to the number of times your code runs. With sub-second metering, AWS Lambda offers a truly cost optimized solution for your serverless environment. AWS records the compute time in milliseconds and the quantity of Lambda functions run to ascertain the cost of the service.
    A. **Event source.** Event sources are AWS services that can be used to trigger your Lambda functions, or put another way, they produce the events that your Lambda function essentially responds to by invoking it.Trigger.
    B. **The Trigger** is essentially an operation from an event source that causes the function to invoke. So essentially triggering that function. For example, an Amazon S3 put request could be used as a trigger. the trigger is an operation from an event source that causes the function to invoke and in my previous statement,
    C. **Downstream Resources.** These are the resources that are required during the execution of your Lambda function. For example, your function might call upon accessing a specific SNS topic, or a particular SQS queue. So they are not used as the source of the trigger, but instead they are the resources to be used to execute the code within the function upon invocation.
    D. **Log streams.** In an effort to help you identify issues and troubleshoot issues with your Lambda function, you can add logging statements to help you identify if your code is operating as expected into a log stream. These log streams will essentially be a sequence of events that all come from the same function and recorded in CloudWatch. In addition to log streams, Lambda also sends common metrics of your functions to CloudWatch for monitoring and alerting.
64. **AWS Batch:** this service is used to manage and run Batch computing workloads within AWS. All provisioning, monitoring, maintenance and management of the clusters themselves is taken care of by AWS, meaning there is no software to be installed by yourself.

65. AWS Batch does not manage all batch computing environments. Depending on the type of compute environment you select, you may manage the infrastructure. It does not apply a stream processing model, and it does not combine the power of all instances deployed in a VPC network.
    A. **Jobs.** A job is classed as a unit of work that is to be run by AWS Batch. For example, this can be a Linux executable file, an application within an ECS cluster or a shell script. The jobs themselves run on EC2 instances as a containerized application. Each job can at any one time be in a number of different states, for example, submitted, pending, running, failed, among others.
    B. **Job definitions.** These define specific parameters for the jobs themselves. They dictate how the job will run and with what configuration. Some examples of these may be how many vCPUs to use for the container, which data volume should be used, which IAM role should be used, allowing access for AWS Batch to communicate with other AWS services, and mount points.
    C. **Job queues.** Jobs that are scheduled are placed into a job queue until they run. **It's also possible to have multiple queues with different priorities if needed.** One queue could be used for on-demand EC2 instances, and another queue could be used for the spot instances. Both on-demand and spot instances are supported by AWS Batch, allowing you to optimize cost, and AWS Batch can even bid on your behalf for those spot instances. When you submit an AWS Batch job, you submit it to a particular job queue, where it resides until it is scheduled onto a compute environment. You associate one or more compute environments with a job queue, and you can assign priority values for these compute environments and even across job queues themselves. For example, you could have a high priority queue that you submit time-sensitive jobs to, and a low priority queue for jobs that can run anytime when compute resources are cheaper.
    D. **Job scheduling.** The Job Scheduler takes control of when a job should be run and from which Compute Environment. Typically it will operate on a first-in-first-out basis, and it will look at the different job queues that you have configured, ensuring that higher priority queues are run first, assuming all dependencies of that job have been met.
    E. **Compute Environments.** These are the environments containing the compute resources to carry out the job. The environment can be defined as managed or unmanaged. A managed environment means that the service itself will handle provisioning, scaling and termination of your Compute instances based on the configuration parameters that you would enter regarding the instance type, purchase method, such as on-demand or spot. This environment is then created as an Amazon ECS Cluster. Unmanaged environments are provisioned, managed and maintained by you, which gives greater customization. However, it does require greater administration and maintenance and also requires you to create the necessary Amazon ECS Cluster that the managed environment would have done on your behalf. It removes the need for expensive hardware, and time-consuming administrative and process management requirements.
66. If you have a requirement to run multiple jobs in parallel using Batch computing, for example, to analyze financial risk models, perform media transcoding or engineering simulations, then AWS Batch would be a perfect solution.
67. **Amazon Lightsail:** Amazon Lightsail is essentially a virtual private server, A VPS, backed by AWS infrastructure, much like an EC2 instance but without as many configurable steps throughout its creation.
68. It has been designed to simple, quick, and very easy to use at a low cost point for small-scale use cases by small business or for single users. With its simplicity and small-scale use, it's commonly used to host simple websites, small applications, and blogs. You can run multiple Lightsail instances together, allowing them to communicate. And it's even possible if required to connect it to other AWS resources and to your existing VPC, running within AWS via a peering connection.
69. The instances are charged as an on-demand price, so you'll only pay for the resource when you're using them. The dollar per month price is based on having the instance on continuously, which AWS calculates as 31.25 days multiplied by 24 hours.
    A. **Connect.** This option allows you to connect to your newly created instance using SSH either via inline SSH software provided by Lightsail or with your own SSH software using the key pair provided. The instance is given a public IP to allow you to connect.
    B. **Storage.** This provides an overview of your current storage, showing the capacity and the disk path. For example, /dev/sda1. You also have the ability to attach additional disks to your instance.
    C. **Metrics.** This allows you to view graphical metrics of your instance, such as CPU utilization, network in, network out, StatusCheckFailed, StatusCheckFailed_Instance, and StatusCheckFailed_System. These graphs can be viewed over a number of different time periods, from one hour through to two weeks.
    D. **Networking.** The networking tab allows you to view your IP address information along with a very simple virtual file, allowing you to control which ports your instance can accept connections from. You can also gain additional information on load balancing your traffic between instances.
    E. **Snapshots.** This provides a simple way to backup your instance. Tags. Here you can configure additional or edit existing tags to help you filter and organize your resources. Key-value tags can also be used to help manage your billing and control access.
    F. **History.** This provides simple order information of your instance, such as the date and time the instance was created or when configuration changes occurred.
    G. **Delete.** When you have finished with your instance, this tab allows you to delete your instance along with any data that was stored in it.
70. **Auto Scaling & Elastic Load Balancing:**
71. Auto Scaling keeps checking the health of the EC2 instances launched by it at regular intervals. If an instance is observed as unhealthy, Auto Scaling will automatically terminate the instance and launch a new healthy instance. Thus, it maintains

the number of instances as per the Auto Scaling group configuration.

72. **Scaling policies** help you to adjust the capacity of a group of instances within a fleet by creating a policy for increasing the group size and a policy here for decreasing the group size. It does not allow you to create a policy for each individual instance, provide preconfigure default scaling policies, or schedule fleet health checks.
73. The primary purpose of the launch template when configuring an auto-scaling group is to build a standard configuration to launch instances for your auto-scaling groups. Configuring storage volumes is a general function of auto-scaling, and identifying the best instance type and configuring security groups within an existing auto-scaling group are not primary purposes for using the launch template.
74. You are configuring your application's compute layer using EC2 Auto Scaling. When configuring the group's capacity, you have set the auto scaling group's minimum capacity to four, the desired capacity to 8, and the maximum capacity to 16, When you deploy your auto scaling group, and the instances are deployed, 8 instances are in your auto scaling group.
75. A user has configured an Auto Scaling group with the minimum capacity of three (3) instances, and the maximum capacity of ten (10) instances. You have not specified the desired capacity. When the auto scaling group's configuration is complete, 3 instances will be launched.
76. You configure the size of your Auto Scaling group by setting the minimum, maximum, and desired capacity.
    A. The minimum and maximum capacity are required to create an Auto Scaling group, while the desired capacity is optional. If you do not define your desired capacity upfront, it defaults to your minimum capacity.
    B. By default, the minimum, maximum, and desired capacity are set to one instance when you create an Auto Scaling group from the console. If you change the desired capacity, the capacity that you specify will be the total number of instances launched right after creating your Auto Scaling group.
    C. An Auto Scaling group is elastic as long as it has different values for minimum and maximum capacity. All requests to change the Auto Scaling group's desired capacity (either by manual scaling or automatic scaling) must fall within these limits.
    D. If you choose to automatically scale your group, the maximum limit lets Amazon EC2 Auto Scaling scale out the number of instances as needed to handle an increase in demand. The minimum limit helps ensure that you always have a certain number of instances running at all times.
    E. When the user configures the launch configuration and the Auto Scaling group, the Auto Scaling group will start instances by launching the minimum number (or the desired number, if specified) of EC2 instances. If there are no other scaling conditions attached to the Auto Scaling group, it will maintain the minimum number of running instances at all times.
77. **An internal ELB** serves requests from within the VPC only. It does not serve requests from the Internet-facing target group, as that would be accomplished by the public ELB. It does not serve requests from configured Availability Zones, as the Availability Zone for the ELB node must be configured or it will not route traffic in response to any requests, and it does not necessarily respond to requests from a single EC2 instance, as the idea of elastic load balancing is to balance loads across target groups.
78. The primary role of a **server certificate** in configuring HTTPS as a listener is to terminate the encrypted connection received from the client and then decrypt and forward it to the resources in the ELB target group. The primary role of the certificate is not to identify resources in the Target Group for the connection request. If you have used a certificate created outside AWS, you should upload a third-party IAM certificate; however, the server certificate does not primarily serve the purpose of importing a certificate created outside of AWS.
    A. **the Application Load Balancer** provides a flexible feature set for your web applications running the HTTP or HTTPS protocols. The Application Load Balancer operates **at the request level,** and it also provides advanced routing, TLS termination, and visibility features **targeted at application architectures**, allowing you to route traffic to different ports on the same EC2 instance. This type of Elastic Load Balancer support is ideal for receiving inbound traffic from the clients outside the VPC, offers TLS termination, and advanced routing.
    B. **Network Load Balancers** are used for ultra-high performance for your application while at the same time managing very low latencies. It operates **at the connection level,** routing traffic to targets within your VPC, and it's also capable of handling millions of requests per second.
    C. **Classic Load Balancers** are primarily used for applications that were built in the existing EC2 Classic environment and operate **at both the connection and request level.**
79. A **Target Group** is a group of resources to which an ELB can route requests. It is not a function for routing inbound connections based on ports and protocols set as conditions, a health check that is performed against the resources defined within the target group, or a rule that defines how an incoming request gets routed to a target group.
80. **AWS Outposts:** AWS Outposts service provide AWS server hardware that customers manage on-premises. With AWS Outposts, it's now possible to bring the AWS cloud to your data center. This includes the same hardware used by AWS within their data centers. By bringing in AWS hardware to your data center, it allows you to use native AWS services, including the same tools and APIs as you would when running your infrastructure within AWS, the difference being is that the hardware and services will be running locally to help you maintain the need for local applications and workloads, et cetera.
81. There are two different options available when using Outposts.
    A. **VMware on AWS**, which will seamlessly run your existing VMware management and infrastructure,
    B. **Native AWS variant,** which means you can use the same APIs and management tools as you would in AWS but on-premises.

>>>
The Amazon Storage Service (ASS)
>>

The Amazon Simple Storage Service (Amazon S3)
Amazon Elastic File System (EFS)
Amazon FSx
Amazon Elastic Block Store (EBS)
AWS Storage Gateway
AWS Snow family
AWS DataSync

Block storage. Block storage stores the data in chunks of data known as blocks, and these blocks are stored in a volume, and attached to a single instance. They generally provide very low latency, and can be considered similar to your directly attached disks within your own data center.

File storage. Your data is stored as separate files within a series of directories, forming a data structure hierarchy. The data is then stored on top of a file system, and provides shared access, allowing for multiple users to access the data. File storage in AWS can be associated to your network attached storage systems you may have in your own data center.

Object Storage. Each object does not conform to a data structure hierarchy. Instead, it exists across a flat address space, and is referenced by a unique key. Each object can also have associated metadata to help categorize and identify the object. Now that you have an understanding of why AWS has curated and developed a range of storage services for you to select, let me now start by introducing each of these services to provide information on exactly what the service is and does, and highlighting its key features, and when and why you might select the service.

>>
The Amazon Simple Storage Service (Amazon S3)
>>

-> Amazon S3 is a fully managed, **object-based storage service** that is highly available, highly durable, very cost-effective, and widely accessible.The service operates an object storage service which means each object uploaded does not conform to a data structure hierarchy like a file system would, instead its architecture exists across a flat address space and is referenced by a unique URL.
-> The smallest file size that it supports is **zero bytes** and the largest file size is **five terabytes**. Although there is this size limitation on a maximum file size, it's one that many of us will not perceive as an ongoing inhibitor in the majority of use cases.
-> S3 is a regional service and so when uploading data you as the customer are required to specify the regional location for that data to be placed in. By specifying your region for your data Amazon S3 will then store and duplicate your uploaded data multiple times across multiple availability zones within that region to increase both its durability and availability.
-> **difference between availability and durability is this**:
: when looking at **availability** AWS ensures that the uptime of Amazon S3 is between 99.5% to 99.99%, depending on the storage class, to enable you to access your stored data.
: The **durability percentage** refers to the probability of maintaining your data without it being lost through corruption, degradation of data, or other unknown potential damaging effects.

-> **To store objects in S3,** you first need to define and create a bucket. You can think of a **bucket** as a container for your data. This bucket name must be completely **unique**, not just within the region you specify, but globally against all other S3 buckets that exist, of which there are many millions. And this is because of the flat address space, you simply can't have a duplicate name. Once you have created your bucket you can then begin to upload your data within it. By default your account can have up to a **hundred buckets, but this is a soft limit** and a request to increase this can be made with AWS. Any object uploaded to your buckets are given a unique object key to identify it. In addition to your bucket, you can if required **create folders within the bucket to aid with categorization** of your objects for easier data management. Although folders can provide additional management from a data organization point of view, I want to reiterate that Amazon S3 is not a file system and many features of Amazon S3 work at the bucket level and not a specific folder level and so the unique object key for every object contains the bucket, any folders that are present, and also the name of the file itself.

-> **Versioning** keeps all versions of an object in the same bucket and server access logging logs requests for access to your bucket. You can also use key value pair tags, you can activate object level logging which will record any API activity with CloudTrail associated with your objects and you can also encrypt your objects as well.

>>>
The Amazon Simple Storage Service (Amazon S3) Classes
>>>

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days

->Amazon storage classes:

The storage classes available are as follows
: **S3 Standard -** This storage class is considered a **general-purpose storage class**.
-> It is ideal for a range of use cases where you need high throughput with low latency with the added ability of being able to access your data frequently.
-> By copying data to multiple availability zones, S3 Standard offers eleven nines of durability across multiple availability zones, meaning the OData remains protected against a single availability zone failure.
-> It also offers a 99.99% availability across the year, which is the highest availability that S3 offers.
-> From a security standpoint this storage class also has the added support of SSL, Secure Sockets Layer, for encrypting data in transit in addition to encryption options for when the data is at rest.
-> With management features such as lifecycle rules, objects in S3 Standard can automatically be moved to another storage class. For those unfamiliar with life cycle rules, they provide an automatic method of managing the life of your data while it is being stored on Amazon S3. By adding a life cycle wall to a bucket you are able to configure and set specific criteria that can automatically move your data from one class to another or delete it from Amazon S3 altogether. You may want to do this as a cost saving exercise by moving data to a cheaper storage class after a set period of time.

: S3 Intelligent Tiering - This storage class is ideal for those circumstances **where the frequency of access to the object is unknown**.
-> Effectively, we have **unpredictable data access patterns** and so by using this storage class, it can help to optimize your storage costs.
-> Depending on your data access patterns of objects in the Intelligent Tiering Class, S3 will move your objects between two different tiers, these being **frequent and infrequent access.**
-> Now, these classes are a part of the Intelligent Tiering Class itself and are separate from the existing storage classes I listed earlier.
When the objects are moved to Intelligent Tiering, they are placed within the frequent access tier, which is the more expensive of the two tiers.
If an object is not accessed for 30 days then AWS will automatically move the object to the cheaper tier known as the infrequent access tier. Once that same object is accessed again, it will automatically be moved back to the frequent tier.
-> Much like S3 Standard, S3 Intelligent Tiering also offers 11 nines of durability across multiple availability zones offering protection against the loss of a single AZ. However, its availability isn't quite as high as S3 Standard as it set at 99.9%.
-> This storage class also has the added support of SSL for encrypting data in transit in addition to encryption options for when the data is at rest.
-> S3 Intelligent Tiering also supports the lifecycle rules and matches the same performance throughput and low latency as S3 Standard.

: S3 Standard Infrequent Access - This can be seen as the equivalent to the **infrequent tier from the Intelligent Tiering class** as it is designed for data that does not need to be accessed as frequently as data within the Standard tier, and yet still offers high throughput and low latency access, much like S3 Standard does.
-> As with all other S3 storage classes, it carries that 11 9s durability across multiple AZs, again by copying your objects to multiple availability zones within a single region to protect against AZ outages.
-> It shares the same availability as Intelligent Tiering of 99.9 percent. As a result, this storage class comes at a cheaper cost than S3 Standard.
-> Common security features such as SSL for encryption in transit and data at rest encryption is supported
-> management controls such as lifecycle rules to automatically move objects to an alternate storage class based on your requirements.

: S3 One Zone Infrequent Access - However, again, being an **infrequent storage class it is designed for objects that are unlikely to be accessed frequently.**
-> It also carries the same throughput and low latency. However, the durability, although remaining at eleven nines only exists across a single availability zone.
-> As the name implies to this class it is one zone, as in one availability zone.
-> So the objects will be copied multiple times to different storage locations within the same availability zone instead of across multiple availability zones. This results in a 20% storage cost reduction when compared to S3 Standard.
-> One Zone IA does, however, offer the lowest level of availability which is currently 99.5 percent and this is down to the fact that your data is being stored in a single availability zone.
-> Should the AZ storing your data become unavailable then you will lose access to your data or even worse it may become completely lost should the AZ be destroyed in a catastrophic event.

-> life cycle rules and encryption mechanisms are in place to protect your data both in transit and at rest.

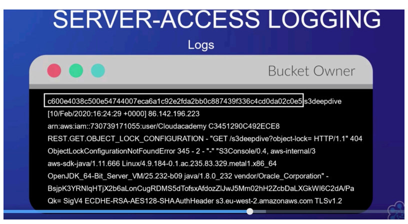
: S3 Glacier and : S3 Glacier Deep Archive - The next two storage classes are associated with **S3 Glacier which is used for archival data**.
-> S3 Glacier, as it can be accessed separately from the Amazon S3 service but closely interacts with it S3 Glacier storage classes directly interact with the Amazon S3 lifecycle rules.
-> However, the fundamental difference with the Amazon Glacier storage classes come at a fraction of the cost when it comes to storing the same amount of data than the S3 storage classes as it doesn't provide you the same features as Amazon S3 but more importantly, **it doesn't provide you instant access to your data**.
-> They offer an **extremely low-cost long term durable storage solution** which is often referred to as **cold storage,** ideally suited for long term backup and archival requirements.
-> It's capable of storing the same data types as Amazon S3, effectively any object, however, it doesn't provide instant access to your data.
-> The service itself has **11 nines of durability** making this just as durable as Amazon S3. Again this is achieved by replicating the data across multiple different availability zones within a single region
-> it provides the storage at a considerably lower cost compared to that of Amazon S3. And this is because **retrieval of data stored in Glacier is not an instant access retrieval process.** When retrieving your data it can take up to several hours to gain access to it depending on certain criteria.
-> The data structure within Glacier is centered around vaults and Archives. **Buckets and folders are not used.** They are purely used for S3.
-> **A Glacier vault** simply acts as a **container for Glacier archives**. These vaults are regional and as such during the creation of these vaults, you are asked to supply the region in which they will reside.
-> Within these vaults, we then have our data which is stored as an archive and these archives can be any object similar to S3.
-> you can have unlimited archives within your Glacier vaults, so from a capacity perspective, it follows the same rule as S3. Effectively you have access to an unlimited quantity of storage for your archives and vaults.
-> Now whereas Amazon S3 provided a nice graphical user interface to view, manage, and retrieve your data within buckets and folders, Amazon Glacier does not offer this service.
-> **The Glacier dashboard within AWS management console allows you to create your vaults, set data retrieval policies, and event notifications.**
-> When it comes to moving data into S3 Glacier for the first time it's effectively a two-step process.
: Firstly, you need to create your vaults as your container for your archives and this could be completed using the Glacier console.
: Secondly, you need to move your data into the Glacier vault using the available API or SDKs.
-> there's also another method of moving your data into Glacier and this is by using the S3 lifecycle rules.
-> When it comes to retrieving your archives, which is your data, you will again have to use some form of code to do so, either the APIs, SDKs or the AWS CLI. Either way, you must first create an archival retrieval job, then request access to all or part of that archive.
-> two S3 Glacier storage classes.
: **S3 Glacier.** This is the default Standard storage class within S3 Glacier offering a highly secure using in transit and at rest encryption low-cost and durable storage solution.
-> The durability matches that of other S3 storage classes, being 11 9s across multiple availability zones, and the availability of S3 Glacier is 99.9%.
-> It's simple to add data to this storage class using the S3 put APIs is in addition to S3 lifecycle rules. However, it does offer a variety of retrieval options depending on how urgently you need the data back, each offering a different price point. These being **expedited, Standard, and bulk.**
: **Expedited.** This is used when you have an urgent requirement to retrieve your data but the request has to be less than 250 megabytes. The data is then made available to you in one to five minutes and this is the most expensive retrieval option of the three.
: **Standard.** This can be used to retrieve any of your archives no matter their size but your data will be available in three to five hours, so much longer than the expedited option and this is the second most expensive of the three options.
: **bulk.** This option is used to retrieve petabytes of data at a time, however, this typically takes between five and twelve hours to complete. This is the cheapest of the retrieval options so it really depends on how much data and how quickly you need it, as the retrieval speed and cost to be made by your retrieval option.
: S3 Glacier Deep Archive. Out of all the storage classes offered by S3,
-> is an ideal storage class for circumstances that require specific data retention regulations and compliance with minimal access, such as those within the financial or health sector where data records might need to be legally retained for seven years or even longer.
-> The durability and availability matches that of S3 Glacier with eleven 9s durability across multiple AZss with 99.9% availability.
-> Adding data into deep archive follows the same processes as S3 Glacier, using S3 put APIs in addition to S3 lifecycle rules.
-> Deep Archive, however, does not offer multiple retrieval options. Instead, AWS states that the retrieval of the data will be within 12 hours or less.
.

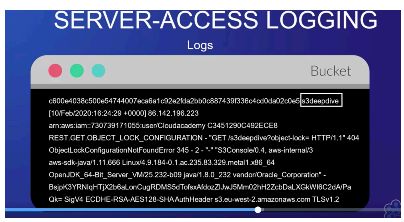
>>>
The Amazon Simple Storage Service (Amazon S3) Versioning:
>>>

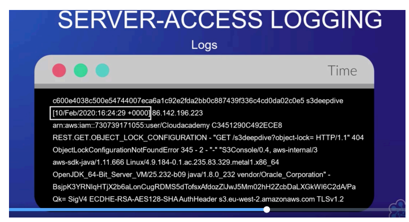
-> This is a bucket feature that allows for multiple versions of the same object to exist. This is useful to allow you to retrieve previous versions of a file, or recover a file should it subjected to accidental deletion, or intended malicious deletion of an object.
-> Versioning is not **enabled by default**, however, **once you have enabled it, you can't disable it**, instead, **you can only suspend it on the bucket which will prevent any further versions from being created of your objects, but it will keep all existing versions of objects up to the point of suspension**
-> With versioning enabled you should also bear in mind that you will incur additional storage costs as you will be storing multiple versions of the same file, and one of the costing metrics of S3 is how much data storage you use.
-> delete marker version becomes the current version of the object and ensures that any GET request to access the object will return a '404 not found value'. However, ALL other versions still exist, including the same version of the file that was submitted for deletion, so you can still view and open these files from the console when you have all versions shown, or by specifying the VersionID in a GET API call to call a specific version.
-> If you wanted to permanently delete an object in a versioned bucket then you must use the **DELETE** Object versionId in an AWS SDK, specifying the exact version that you want to delete. If you delete the version with the delete marker, then the object would reappear in your bucket using the latest version of the object available.
-> buckets can be in the form of any one of these 3 states:

Unversioned (this is the default state for buckets),
Versioning-enabled:
versioning-suspended.

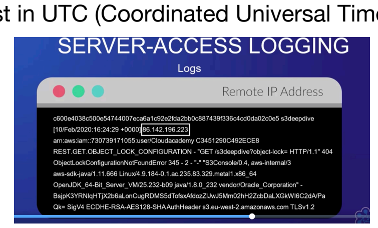
>>>
The Amazon Simple Storage Service (Amazon S3) Server-access logging:
>>>
-> when **server-access logging** is enabled on a bucket it captures details of requests that are made to that bucket and its objects.
-> If the configuration of your access logging is configured using the management console, then the enablement of logging automatically adds the Log Delivery group to the ACL (Access Control List) of the target bucket, allowing the relevant access. However, if you were to configure the access logging using the S3 API or AWS SDKs, then you would need to manually configure these permissions manually.
-> Firstly, and as I've already mentioned, both the source and target buckets should be in the same region, and it's a best practice that different buckets are used for each.
-> Also, the permissions of the S3 Access Log Group can only be assigned via Access Control Lists and not through bucket policies, so when manually setting the permissions for this via an SDK, you must update the ACL.
-> Finally, if you have encryption enabled on your target bucket, access logs will only be delivered if this is set to SSE-S3 (Server-side encryption managed by S3) as encryption with KMS (Key Management Service) is not supported.
-> When the logs start arriving in the target bucket, the names will be presented following a standard naming patter

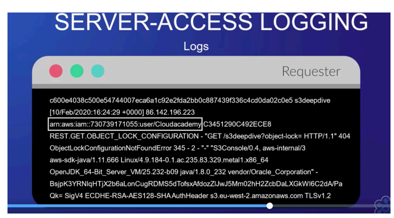




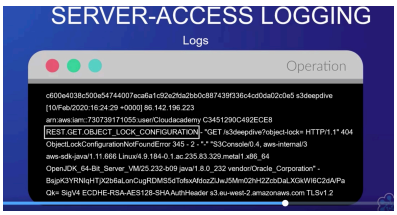


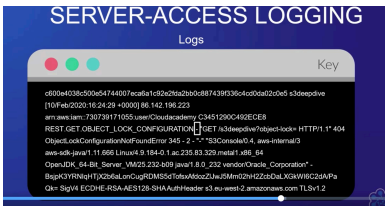
Bucket owner - Represents the canonical user ID of the owner of the Source bucket. The canonical user ID is used for cross-account access via bucket policies.
Bucket - This shows the name of the bucket related to the request.
Time - This is a timestamp of the request in UTC (Coordinated Universal Time).

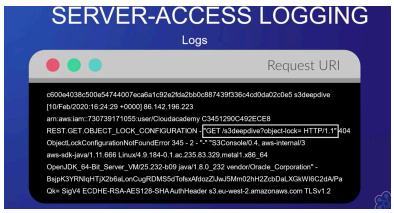




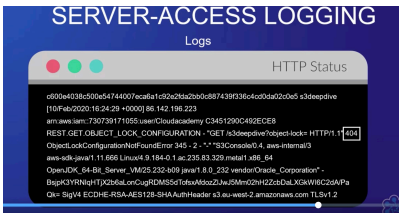
Remote IP Address - Represents the internet address of the identity carrying out the request.
Requester - For authenticated users, this field will show the IAM identity. For any unauthenticated users a hyphen (-) would be displayed instead.
Request ID - A random string to identify each request.

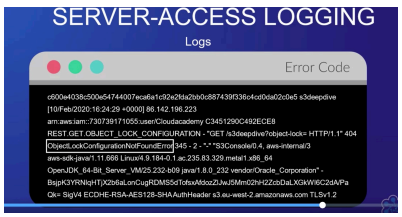


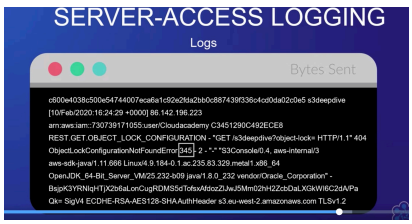




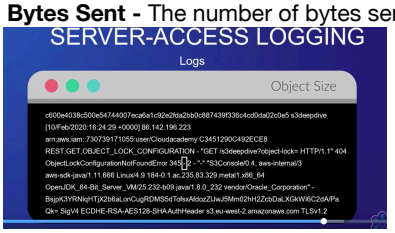
Operation - This will display the operation of the request that was carried out.
Key - The "key" part of the request, URL encoded, or if no key parameter is used then a hyphen will be displayed as in this example. A hyphen in any field of the request indicates that the available data was not known or was not applicable for the request.
Request URI - This represents the Request-URI element of the HTTP request.

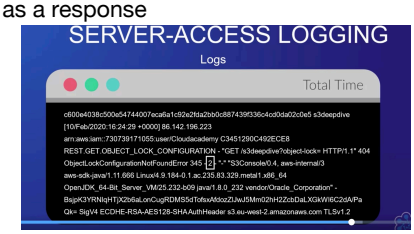


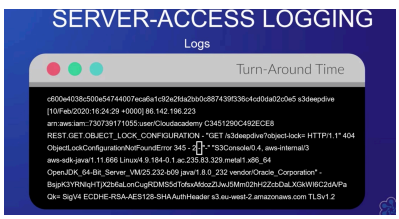




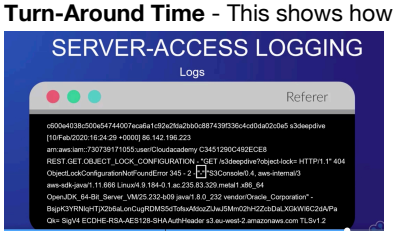
HTTP Status - This displays the HTTP status returned from the request as a numeric value.
Error Code - If an error was experienced, then S3 will return the error code received.
Bytes Sent - The number of bytes sent as a response

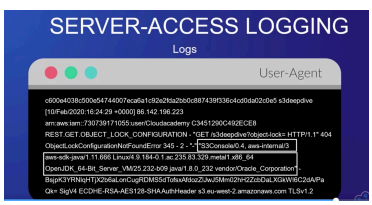


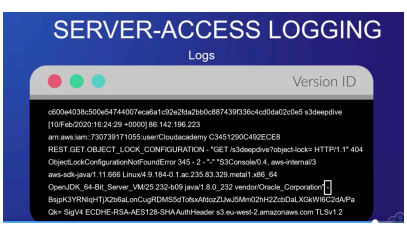




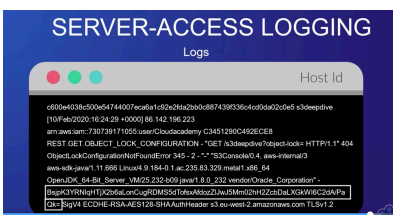
Object Size - The size of the object in question in the request.
Total Time: - Measured in milliseconds, it represents how long the request took from receiving the request to the last byte of sending a response.
Turn-Around Time - This shows how long it took S3 to process the request.

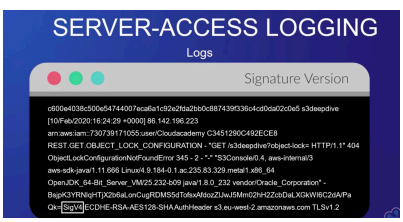


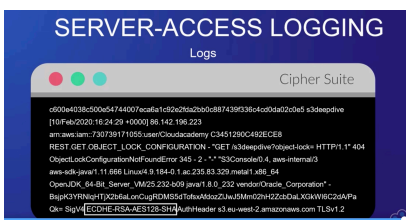




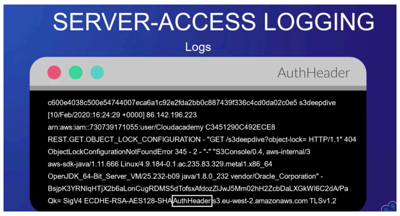
Referer - The value is taken from the HTTP referer header, however, in this case, there was none present and so a hyphen is represented as the value.
User-Agent - This shows the value taken from the HTTP user-agent header.
Version ID - If present, this will show the Version ID of the request.

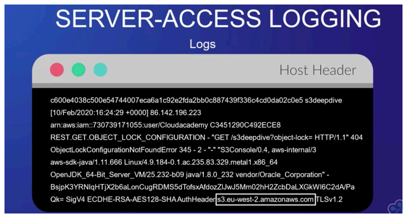


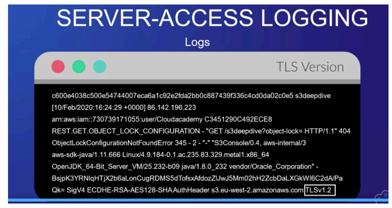




Host Id - The x-amz-id-2 or Amazon S3 extended request ID. The x-amz-id-2 header is a token that is used together with the x-amz-request-id header to help AWS troubleshoot problems.
Signature Version - This will show which signature version was used to authenticate the request.
Cipher Suite - If SSL was used it will show which cipher suite was used. If HTTP was used, then a hyphen would be shown instead.







Authentication Type - This shows the type of authentication used for the request.
Host Header - Represents the endpoints used to connect to Amazon S3 in the request.
TLS Version - This shows which version of TLS was used by the client.

-> Logging is important when it comes to security, root-cause cause analysis following incidents, and it can also be required to conform to specific audit and governance certifications.
-> Server-access logging, however, is not guaranteed and is conducted on a best-effort basis by S3.
-> The logs themselves are collated and sent every few hours, or potentially sooner. There is no hard and fast rule that dictates that every request will be captured and that you will receive a log for a specific request within a set time frame.
-> To allow S3 to write access logs to a target bucket, it will, of course, require specific permissions. These permissions will require write access for a group known as the Log Delivery group, which is a pre-defined Amazon S3 group used to deliver log files to your target buckets.

>>
S3 Bucket configurations [Static website Hosting]:
>>

-> S3 Bucket configurations [Static website Hosting]:
-> By default, static website hosting is disabled. When this tile is selected you are presented with three options, Use this bucket to **host a website, Redirect requests, and Disabled.**
-> In addition to these options, there is also a **region-specific website endpoint** shown for your bucket. This endpoint allows users **to access your website via that URL.** However, there a couple of points to understand when using your S3 endpoint as your website URL address.
: Firstly, **it does not support HTTPS requests.**
: **The bucket and its contents must be marked as publicly accessible.**
: And **it does not support Requester Pays.**

-> If you select the **option of Use this bucket to host a website** you need to provide additional parameters in its configuration.
: Firstly, you need to **add an index document**.
: And your index document will be the default, or home page of your static website.
: The error document will be the page that is displayed when an error occurs. And these documents must be located within your bucket.
-> The **redirection rules** allow you to use XML to create advanced redirect requests to specific content.

-> The other option available to you is **Redirect Requests.** And this option allows you to redirect all traffic to your website endpoint.
-> The target destination could also be another bucket configured for static website hosting. The Protocol field allows you to enter which protocol should be used during the redirect.

-> For people to be able to access your website hosted within your S3 bucket, then the bucket must be accessible to the public, so we need to ensure that the permissions are set correctly. **By default your S3 buckets are blocked to the public.** If you select your bucket and then permissions tab, you will see the current settings for your bucket under the Block all public access.

-> Once your bucket is publicly readable, you will then need to add a bucket policy to allow the public to read your objects within your bucket. This policy allows everyone to have the action of s3:GetObject from your bucket hosting your website And you will be notified via a warning that this will make all objects publicly accessible, accept the warning as this is the outcome that you are trying to configure.

-> to test it using your website endpoint. It is also possible to set up your static website with your own customized domain name instead of the automatically generated website endpoint created by S3. In addition to serving your content via CloudFront.

-> we've unblocked all public access, we've entered a bucket policy that allows anyone to access this bucket using the s3:GetObject action, we've added our HTML file to the bucket, and we've also enabled static website hosting.

>>>
The Amazon Simple Storage Service (Amazon S3) Object level Logging :
>>>
-> This feature is actually more closely related to the **AWS CloudTrail service** than S3 in a way, as it's AWS CloudTrail that performs logging activities against Amazon S3 data events. These data events are specific API calls used in S3, such as **GetObject, DeleteObject, and PutObject.**

-> **CloudTrail** is a **service that has a primary function to record and track all AWS API requests made.** These API calls can be programmatic requests initiated from a user using an SDK, the AWS command-line interface, from within the AWS management console or even from a request made by another AWS service.

-> When an API request is initiated, AWS CloudTrail captures the request as an event and records this event within a log file which is then stored on S3. Each API call represents a new event within the log file. CloudTrail also records and associates other identifying metadata with all the events. For example, the identity of the caller, the timestamp of when the request was initiated and the source IP address.

->Capturing S3 data events can be configured in 2 ways:

Firstly, if you want to capture data events for all or some of your S3 buckets, then you can configure this from within one of your Trails using the **AWS CloudTrail console** itself.

Secondly, if it's not already enabled via **AWS CloudTrail for your bucket** you can configure it at the bucket level using the Properties tab. Selecting the Object-level logging tile will present you with options to configure it.

>>
The Amazon Simple Storage Service (Amazon S3) Object Lock :
>>>
-> Object lock property which is considered an 'advanced' property of an S3 bucket.

-> This feature is often used to meet a level of compliance known as **WORM, meaning Write Once Read Many**. It allows you to offer a level of protection against your objects in your bucket and prevents them from being deleted, either for a set period of time that is defined by you or alternatively prevents it from being deleted until the end of time!
-> The ability to add retention periods using Object Lock help S3 to comply with regulations such as FINRA, the Financial Industry Regulatory Authority.
-> Setting Object Lock on a bucket can only be achieved at the time of the creation of the bucket. If you attempted to enable it on an existing bucket by clicking on the Object Lock tile in the bucket properties, you would receive the following error.
-> To enable and configure object lock during the creation of the bucket, **you first need to ensure that you have Versioning enabled.** Without first enabling versioning, **it is NOT possible to enable object lock, which can be found under the 'Advanced' setting of Step 2 'Configure Options' during creating your bucket.**
-> **Once you have created your bucket with object lock enabled it will be permanently enabled and can't be disabled.**
-> Although your bucket is now configured for 'object lock', any object your place into it at this stage is NOT automatically protected, to ensure they are you need to enable some default options on the bucket first.
-> When you select the Object-lock tile, which will now say **'Permanently enabled.'** You will be presented with two retention modes, and the settings selected here will define the default retention of an object when it is added to the bucket and therefore applying the required protection that object lock provides.

-> These retention modes are **Governance Mode and Compliance Mode.**

By **enabling Governance Mode** it **prevents your users from performing a delete or an overwrite of any of the versions of your objects in the bucket throughout the duration set by the retention period.**

-> However, **if you have very specific permissions, including s3:BypassGovernanceMode, s3:GetObjectLockConfiguration, s3:GetObjectRetention, then a user will still be able to delete an object version within the retention period or change any retention settings set on the bucket.**

-> When **setting Governance Mode** you will be asked to add a retention period in days and therefore defines how long the object is protected by object lock preventing it from being deleted. When an object is added to the bucket, a timestamp is added to the metadata reflecting the retention period. When the retention period is over, the object can then be deleted again.

Compliance Mode. The key difference between Compliance Mode and Governance Mode is that there are NO users that can override the retention periods set or delete an object, and that also includes your AWS root account which has the highest privileges. Essentially, any object added to a bucket configured for Compliance Mode means that the object will remain for the duration of the retention period. Again, much like with Governance Mode, you will be asked to enter a retention period based upon a number of days.

-> You can also set object-lock **on a per-object by object basis** if you didn't want to set a default retention mode of Governance or Compliance. To do so, you need to select the object-lock option of the object's properties itself. Again, you can set either the **governance or compliance retention mode** for that specific object. The 'Retain until date' shows that this object is already bound by a retention mode with a retention period, and as a result, it shows the date in which this object is to be protected until. When this date has passed, the object is no longer protected and can be deleted.

-> **The legal hold** element only appears for object versions and not at the bucket level and acts much like a retention period and prevents the object from being deleted, however, legal holds do not have an expiration date. Therefore, the object will remain protected until a user with permissions of s3:PutObjectLegalHold disables the legal hold on the object. If an object is already protected by a **retention period,** a legal hold can also be placed on the object. When the retention period expires, the object will still be protected by the legal hold regardless of the fact that the retention period has expired.

>>
The Amazon Simple Storage Service (Amazon S3) Tags :
>>

-> the use of cost allocation tags against your S3 buckets.

-> It is likely when using Amazon S3 that you are using it for a variety of different use cases and solutions, across multiple business units and departments, each with different cost centers. This can make it difficult to manage budgets across your organization. **Using bucket tags, known as S3 cost allocation tags, you can assign key-value pairs at the bucket level to help with categorization.**
-> For example, let's suppose you had 3 different buckets each with 2 key-value pairs, Project and Environment. Using these tags, we can see that each bucket belongs to a different 'Project', and also that 2 of them are considered 'Production' and another is 'Test', based on the environment Key.
-> **Tags like this can be used across all AWS services and help you to manage, categorize, and organize your resources in a variety of ways.**
-> Using the Cost Explorer from within your AWS Billing and Cost Management, you can report on these Key values, for example, you could identify and highlight the costs associated with your resources that were tagged with the project, CloudAcademy. This will highlight all the AWS resources that had this key value pair allowing you to get a full understanding of the project costs for that particular project, in this case, 'CloudAcademy'.
-> One point to note is that you must activate your cost allocation tags from within AWS Billing before they will show up on any reports.

>>
The Amazon Simple Storage Service (Amazon S3) Transfer Acceleration :
>>
-> how you can speed up your long-distance S3 data transfers using Transfer Acceleration.
-> When transferring data into or out of Amazon S3 from and to your remote client, or to another AWS region, transfer acceleration can dramatically speed up the process by utilizing another AWS service, **Amazon CloudFront.**
-> **Amazon CloudFront is a content delivery network service (CDN),** which essentially provides a means of distributing traffic worldwide via edge locations. AWS edge locations are sites deployed in major cities and highly populated areas across the globe. More information on Amazon CloudFront can be found in our existing course here.
->When transferring data to S3 from your client with transfer acceleration enabled at the bucket level, the request will go via one of the CloudFront Edge Locations, from here the transfer request will then be routed through a high speed optimized AWS network path to Amazon S3.
-> When using transfer acceleration you should be aware that there is a cost. Whereas normal data transfer into amazon S3 is free from the internet, with transfer acceleration, this is a cost associated per GB depending on which edge location is used. Also, there is an increased cost for any data transferred OUT of S3, either to the internet or to another Region, again due to the edge location acceleration involved.
-> to enable transfer acceleration your bucket name must be DNS compliant and not contain any periods at all. Also, to make use of the transfer acceleration feature itself, any requests, such as GET or PUT to the bucket, must use this new transfer acceleration endpoint.
-> there are a couple of S3 operations that it does not support, these being: GET Service (list buckets), PUT Bucket (create bucket), DELETE Bucket, and Cross region copies using PUT Object - Copy.

>>
The Amazon Simple Storage Service (Amazon S3) Requester Pays :
>>

-> an advanced property of an S3 bucket, Requester Pays.
-> As the name implies, **when this feature is configured any costs associated with requests and data transfer becomes the responsibility of the requester instead of the bucket owner.** The bucket owner will still, however, pay for the storage costs associated with the objects stored in the bucket.
-> A fundamental condition of enabling requester pays is ensuring that all access is authenticated to your bucket, as anonymous access requests will not be able to take advantage of the requester pays attribute. This is because AWS would not know which account to charge the request and data download to. By authenticating requests, it allows a trace back to the identity and to which AWS account that identity is originating from, and the cost is then transferred to that account.

-> To enable requester pays is very simply, select the 'Requester Pays' tile from the properties of the required bucket. The

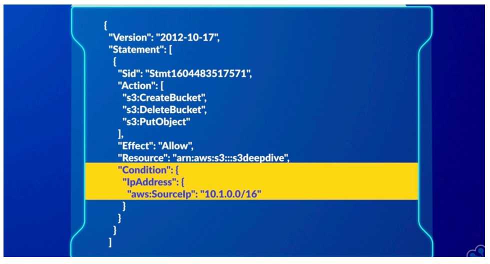
options available are either to enable requester pays or disable requester pays. Once your decision has been made, select Save. From this point, any POST, GET or HEAD requests to the bucket must include **x-amz-request-payer in the header and this parameter confirms that the requester is aware that there are cost implications associated with that request for requester pays.**

>>>
The Amazon Simple Storage Service (Amazon S3) : Different methods of using policies and permissions
>>>

-> different methods of using policies and permissions to gain access to S3 resources, which include both S3 buckets and objects within those buckets. We can control access to S3 resources via policies using both

-> **identity-based policies and resource-based policies.**

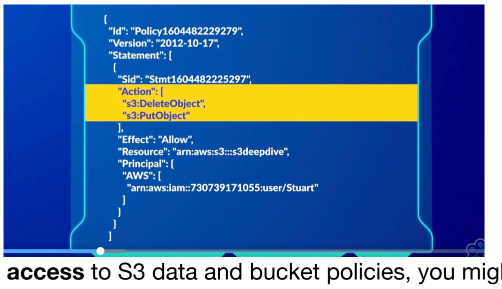
-> **Identity-based policies** are attached to the IAM identity requiring access, and then using IAM permission policies, either in-line or managed, they can then be associated to the user, a group that the user belongs to, or via a role that the user has permission to assume. Identity-based policies define the resource in the policy.



-> For example, the bucket name, as you can see in this example policy shown here. So this identity-based policy can be associated with a user and will permit them to use all S3 actions within the bucket, s3deepdive. You can of course use conditions within your policies to manage and refine access to your resources even further. For example, in this policy, the identity would be able to perform the actions CreateBucket, DeleteBucket, and PutObject on the s3deepdive bucket on the condition that their source IP address was within the range of the CIDR block 10.1.0.0/16.

-> **resource-based policie**s differ in the fact that the policy is associated with a resource rather than the identity. So from an Amazon S3 perspective, resource-based policies come in the form of Access Control Lists and bucket policies. Because of this, you need to define within the policy who will be allowed or denied access. This is managed differently depending on if you're using **bucket policies or Access Control Lists.**

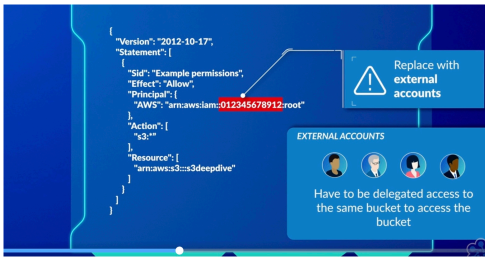
: **bucket policies.** A bucket policy is written in JSON and is directly attached to your bucket as another means of granting and restricting access control. And by default, when you create a bucket, no bucket policy exists. To add a modified bucket policy, you must select the bucket, go to permissions, and then select the Edit button in the bucket policy section. From here, you have one of three different options. You can either freely write your own bucket policy in the JSON window, view some policy examples, or use the AWS policy generator to help you create your bucket policy. When working with bucket policies, you must specify a principle element, which defines the principal who is associated with the action and effect defined in the statement.



-> example bucket policy to make it clearer. This bucket policy allows the actions, s3 DeleteObject and s3 PutObject, for the s3deepdive bucket for the principal Stuart. You'll notice that with the identity-based policy shown previously, I didn't need to add the Principle element. It's important to note that the permissions defined within the statements of a bucket policy apply to the bucket and the objects that reside within the bucket.

-> So now, we are aware that you can use both **IAM policies to control access** to S3 data and bucket policies, you might be wondering when you should use one over the other.

-> You might **want to use IAM if you want to centrally manage your access control methods all in the one service**, that being IAM.
-> Also, **if you have multiple permissions to apply to a large number of buckets, the management of doing so might be easier to do from within IAM access across one or two policies** rather than creating separate bucket policies that you would have to attach to each of your buckets.
-> Also, **IAM has the added advantage of being able to control access for more than one service at a time within its policies**, whereas bucket policies only control access to the S3 bucket and its objects.



-> On the flip side, you might **want to use bucket policies if you want to maintain your security policies within S3 alone**.
-> **You can also grant cross-account access using bucket policies without having to create and assume roles that are created within IAM.** For example, all you'd need to do is to create a bucket policy as shown and attach it to your bucket, for example, s3deepdive, replacing the text in red with the external account. In the external account, users then simply have to be delegated access to the same bucket to be able to access the bucket.

-> Also, another reason you might **want to use bucket policies is because IAM policies can be a maximum of two kilobytes in size for users, five kilobytes for groups, and 10 kilobytes for roles. However, bucket policies can reach a size of 20 kilobytes.**

-> Now, you can of course use both IAM policies and bucket policies to control access. **They are not mutually exclusive.** They can both be used together to control access. I should come to how multiple access controls are evaluated and their logic after we look at S3 Access Control Lists, ACLs, which as I mentioned previously are another resource-based access control method.

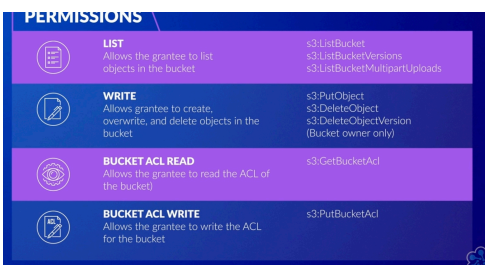
-**> S3 Access Control Lists, or ACLs**, allow you to control access to buckets in addition to specific objects within a bucket by groupings and AWS accounts. One advantage of being able to **use ACLs is that you can set different permissions per object.**

-> ACLs **do not follow the same JSON format as policies defined by IAM and bucket policies.** Instead, they are far less granular, and different permissions can be applied depending **if you are applying an ACL at the bucket level or the object level.**

-> Due to the basic structure of an ACL, **it is not possible to implicitly deny access using ACLs. Neither are you able to implement conditional elements, like we saw earlier when I mentioned identity-based access.** An example of default ACL for a bucket looks as shown.

-> Permissions are based on the grantee of which there are four:
The bucket owner. This'll be your own AWS account and will have full control over all objects and the bucket itself.
Everyone (public access). Permissions set against this grantee would mean anyone can access using the permissions applied, providing the object had been made public. These requests can be signed, authenticated, or unsigned, unauthenticated.
Authenticated users. This option will allow IAM users from any AWS account to access the object via signed request, authenticated.
S3 log delivery group. This is a group used to deliver log files when server access logs has been configured, and the bucket is used to store and write log files to.

-> By editing the ACL at the bucket level, you'll be presented with the different levels of permissions that can be applied to the bucket. It will show you the permissions given to the grantee groups for the bucket, which are either List or Write. You can also see what permissions have been given to enable the grantee access to either read or write against the bucket ACL.



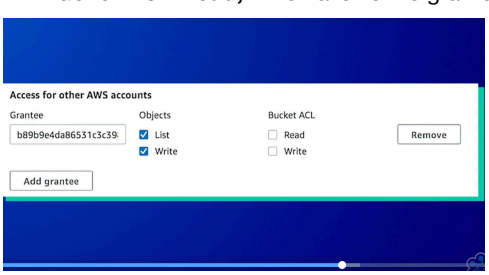
-> For clarification about what these **permissions actually grant,** they are as follows.

List, which allows the grantee to list objects in the bucket.

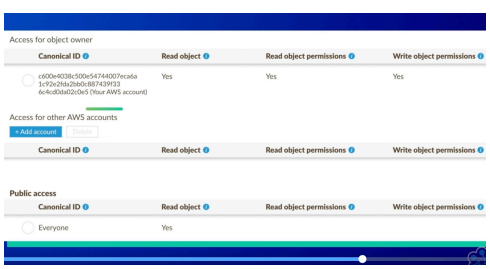
Write, which allows a grantee to create, overwrite, and delete objects in the bucket.

Bucket ACL Read, which allows the grantee to read the ACL of the bucket. And

Bucket ACL Write, which allows the grantee to write the ACL for the bucket.

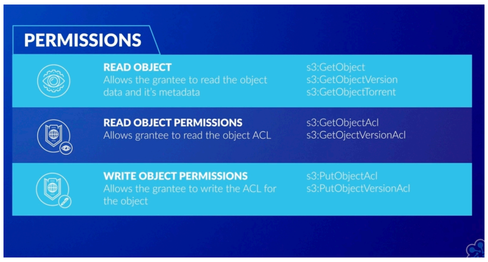


ACL looks like at the bucket level:
-> You may have noticed that you can also add access for another account as a grantee. To configure this access, you'll be required to enter the **canonical ID of the AWS account** that you would like to have access.



ACL looks like at the object level.
-> Here's an ACL of an object within a bucket. As you can see, again, by default, the resource owner has full control over the object. Also, you will notice that you can add access for another account.

-> Again, you'll need to add the appropriate **canonical AWS account ID** and the relevant permissions. You can also specify public access if you have public access enabled, In this example, we can see that the Everyone group has read access to the object.



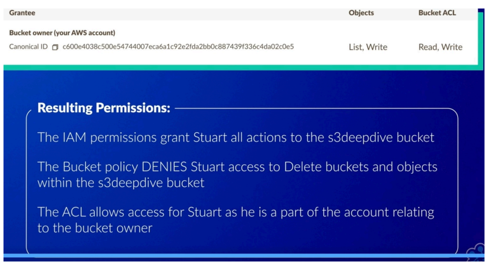
-> there is no write permission when working with object ACLs, unlike bucket ACLs.

->
Read object allows the grantee to read the object data and its metadata.
Read object permissions allows the grantee to read the object ACL.
Write object permissions allows the grantee to write the ACL for the object.

-> IAM policies, S3 bucket policies, and finally S3 Access Control Lists. But what would happen if you used all of these access control methods? How would the policy evaluation logic operate? Which would take precedence?

-> Let's say, for example, someone tries to access an object in a bucket that has S3 ACLs attached in addition to bucket permissions and also have their own IAM permissions. How is this access governed if there are conflicting permissions to the object in the bucket that they are trying to access? All of these policies will be viewed together to determine the resulting access and will handle any permission conflict in accordance with the **principle of least privileged.**

-> **Essentially, by default, AWS states that access is denied to an object, even without an explicit Deny within any policy.**
-> To gain access,
: there has to be an Allow within a policy that the principal is associated to or
: defined by within a bucket policy or ACL.
-> If there was no Deny defined, but there is an Allow within a policy, then access will be authorized.
-> However, if there is a single Deny associated with the principal to a specific object, then even if an Allow does exist, this explicit denial will always take precedence overruling the Allow and access will not be authorized.



-> So going back to our example, we have a user called Stuart who has the following identity-based policy associated. The bucket s3deepdive also has the attached bucket policy. The ACL permissions on the s3deepdive bucket give the bucket owner full control to the bucket and its objects. The user Stuart is a part of the same AWS account as the bucket owner.

-> So what are the resulting permissions? The IAM permissions grant Stuart all actions to the s3deepdive bucket, the bucket policy denies Stuart access to delete buckets and objects within the s3deepdive bucket, and the ACL allows access for Stuart, as he is a part of the account relating to the bucket owner.

=> So as we know, a Deny always takes precedence over an Allow, meaning Stuart will have access to the s3deepdive bucket to perform all S3 actions apart from deleting the bucket or any of its objects.

>>**>**
The Amazon Simple Storage Service (Amazon S3) : Managing public access to the bucket
>>>

-> When creating a new bucket in S3, there's an option that's dedicated to helping you protect your bucket from public access. And by default, you can see that there's a checkbox that's ticked, which blocks all public access.

-> As a result, you have to actively change this setting to allow public access. If you do need some public access to this bucket, then you can turn off the setting, and it allows you to select for additional options that can be used to filter public access.

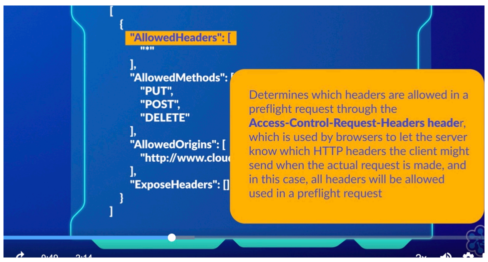
-> So you can :
block public access to buckets and objects granted through new access controllers,
block public access to buckets and objects granted through any access controllers,
block public access to buckets and objects granted through new public bucket or access point policies, and
block public and cross account access to buckets and objects, through any public bucket or access point policy.

-> This allows you to allow some public access based on certain security controls and block others. You don't have to select any or you can have a combination of the four shown. Once you've made your selection, you can review the settings on your bucket by selecting it and viewing the Permissions tab.

-> Because all public access to this bucket is blocked, As a result, if I tried to allow any kind of public or cross account access for the bucket policy or ACL, then access would still not be allowed as the bucket still has the block all public access setting enabled.
-> I don't have permissions to edit these ACL settings, with a response of access denied. And that is because we have the block all public access on. So this overrides the ACL.

-> So as soon as I enabled that block or public access setting, AWS updates all the settings in the bucket and the objects to remove that access.

>>>
The Amazon Simple Storage Service (Amazon S3) : Cross Origin Resource Sharing (CORS)
>>



-> At a high level, **CORS allows specific resources on a webpage to be requested from a different domain than its own.** And this allows you to build client-side web applications. And then if required, you can utilize CORS support to access resources stored in S3.

-> involves the use of policies and these policies are embedded in the CORS configuration of the bucket itself which can be found under the Permissions tab.

-> Example which has a single rule. The following policy allows you to use PUT, POST, and DELETE from the origin of www.cloudacademy.com. The AllowedHeaders element of the policy determines which headers are allowed in a preflight request through the **Access-Control-Request-Headers** header, which is used by browsers to let the server know which HTTP header the client might send when the actual request is made. And in this case, all headers will be allowed to be used in a preflight request.

-> When the bucket receives a preflight request from a browser, S3 will evaluate the policy associated with the bucket for its CORS configuration and will process the first matching rule in the policy. A match is made when the following conditions in the rule are met. The requestor's Origin header matches an entry made in the **AllowedOrigins** element. The method used in the request, for example a POST or DELETE operation is matched in the **AllowedMethods** element. And finally, the headers used within the requests Access-Control-Request-Headers header with a preflight request matches a value in the **AllowedHeader** element.

-> The **ExposeHeader** element in the policy is used to define a header in the response that is allowed to be made by customer applications.

-> Your CORS policy can contain more than one rule.

>>>
The Amazon Simple Storage Service (Amazon S3) : Lifecycle configurations
>>
-> **Lifecycle configurations are an important cost tool that can enable you to delete or transition old unused versions of your objects, clean up incomplete multipart uploads, transition objects to lower cost storage tiers and delete objects that are no longer needed.**

-> Consider the following scenario: you have a data lake workload in a **versioned S3 bucket** that grows at a very fast and consistent pace. Tons of overwrites are occurring daily and large objects are being uploaded via multipart upload. From the data management perspective, you may have incomplete multipart uploads every so often, and tons of out-of-date non-current versions of your data.

-> tools is to manage the storage lifecycle of your data, by moving your data to lower cost storage classes, or deleting data you no longer need. You can, of course, move and delete your data manually, but that can be difficult to manage at scale. So, there are two ways to automate this process:

-> The first way is to **use the S3 Intelligent-Tiering storage class:** You'll pay a monthly object monitoring and automation charge, and in return S3 Intelligent-Tiering will monitor your object access patterns, and automatically move your objects between **three tiers: frequent, infrequent and archival.** S3 Intelligent-Tiering is **recommended for data access patterns that are unknown or unpredictable,** and is meant to give a more **"hands-off" approach to managing your data lifecycle.**

-> The second approach is by **using Lifecycle configurations**. You can use lifecycle configurations **to transition data to**

a lower cost S3 storage class, or to delete data. Lifecycle configurations additionally provide options **to clean up incomplete multipart uploads and manage noncurrent versions of your data** - which ultimately, helps reduce storage spend. Using lifecycle configurations is the **most cost-effective strategy when your objects and workloads have a defined lifecycle and follow predictable patterns of usage.**

-> For example, a defined access pattern may be that you use S3 for logging and only access your logs frequently for at most, a month. After that month, you may not need real-time access, but due to company data retaining policies, you cannot delete them for a year. With this information, you can create a solid lifecycle configuration based on this access pattern.
: You could create an **S3 Lifecycle configuration that transitions objects from the S3 standard storage class to the S3 Glacier Flexible Retrieval storage class after 30 days.** By simply changing the storage class of your objects, you will begin to see significant cost savings in your overall storage spend. And after 365 days, you can then delete the objects and continue to save on costs.

-> You may find that a lot of your data follows a similar access pattern: you slowly stop needing real-time access to your data, and can eventually delete the data after a certain period of time passes. Or you may have data that you need to save to meet some compliance or governance regulation that can be moved from S3 Standard to archival storage and left alone for long periods of time. Or perhaps, you have a ton of objects in S3 Standard storage and you want to transition all of those objects into the S3- Intelligent Tiering storage class. If these patterns sound similar to your use case, then using lifecycle configurations makes sense for your workload.

Components:

-> An S3 Lifecycle configuration is technically an XML file.

-> Each lifecycle configuration contains a set of rules. Each rule is broken up into four components:

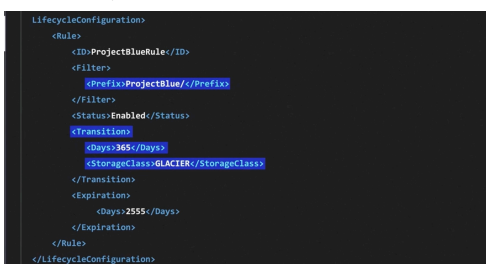
ID: The ID uniquely identifies the lifecycle rule - you can consider this the **name of the rule**. This is important, as one lifecycle configuration can have up to **1000 rules**, and the ID can help you keep track of what rule does what.

Filters: The filters section defines WHICH objects in your bucket you'd like to take action on. You can choose to apply actions to all objects or a subset of objects in a bucket. If you choose a subset of objects, you can filter based on **prefix, object tag, or object size**. Or to be very granular, you could filter based on a combination of these attributes. when you combine multiple filters, you have to use the keyword "And" in XML. **The maximum filter size is 5TB.**

: Example: you can create a filter that transitions all objects with the **prefix** ProjectBlue/, if they also are tagged with the Classification **tag** value "Secret".

For object size, I can transition or expire objects if they're **greater than** a specific size, **less than** a specific size, or if they're in a range between two size bounds. For example, I can create a filter, where I'm transitioning my objects if they're larger than 500 Bytes, but smaller than 10,000 Bytes.

Status: With the status field, you can **enable and disable each lifecycle rule**. This can be helpful when you're testing lifecycle configurations out, as you figure out what the best rules for your workload are. Once you change the status to "disabled", S3 won't run the actions defined in that rule - essentially stopping the lifecycle action. And when you're ready to run those actions on your objects again, you can always change the status back to enabled.

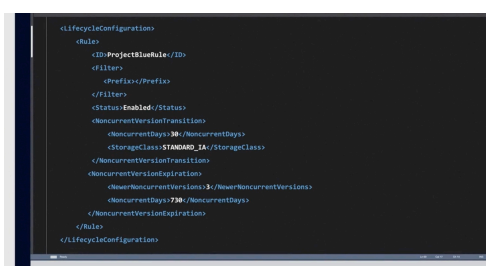


Actions. This is where you define WHERE you want your objects to move to - are you transitioning them to another storage class or are you deleting them. There are six main actions that you can use:

Transition: Transition actions enable you to **move** data automatically between S3 storage classes. you can define when to move these objects based on object age. And age is based on when the object was last created or modified.

expiration: Expiration actions enable you to automate the **deletion** of your objects in S3. you can define when to delete these objects based on object age. And age is based on when the object was last created or modified.

Example: the first action transitions all objects that are prefixed with ProjectBlue/ to the **S3 Glacier Flexible Retrieval storage class** 365 days after they were created. The second action says after 2,550 days - which is 7 years, delete the objects prefixed with ProjectBlue/ because they aren't needed any longer.



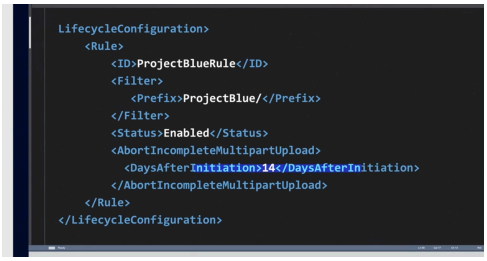
NoncurrentVersionTransition: If you have versioning turned on for your bucket, transition actions and expiration actions only work for the current version of your object. If you want to transition noncurrent versions of your object, you must use the NoncurrentVersionTransition action.

NoncurrentVersionExpiration: if you want to delete noncurrent versions of your object, you must use the NoncurrentVersionExpiration

action
-> you can define when to move or delete objects based on two things.

First is the **number of days since the object has been noncurrent**, which Amazon calculates as the number of days since the object was overwritten or deleted.

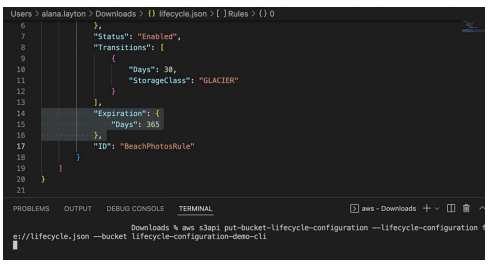
second, is the **maximum number of versions to retain.** This is helpful when you want to save a few versions to rollback to for data protection, while removing old versions of your object to save on storage spend.



Example: In this rule, all noncurrent versions are moved to **S3 Standard - Infrequent Access** 30 days after they become noncurrent. In addition, it deletes all noncurrent versions 730 days, or two years, after they become noncurrent, while retaining the 3 latest versions.
-> you can choose to retain any number of versions between 1 and 100.

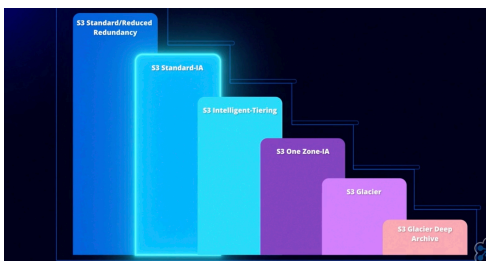
ExpiredObjectDeleteMarker: One of them is the Expired object delete marker action. This is helpful if you have objects that have zero versions, that only have a delete marker left. This is referred to as the expired object delete marker, and you can use this action to remove them.

AbortIncompleteMultipartUpload: If you have incomplete multipart uploads that you need to clean up, you should use this action. With this action, you can specify the maximum time, in days, that your multipart uploads can remain in progress.



Example: you can specify that your multipart uploads can remain in progress for 14 days. If the upload does not complete in 14 days, S3 will delete it.

-> S3 Glacier Flexible Retrieval has a 90 day retention period,



-> configuring lifecycle configurations using the CLI is that you have to provide a JSON template file defining your configuration.

: S3 Limitations and Considerations

S3 Standard storage is at the top of the staircase,
while **S3 Glacier Deep Archive** is at the bottom of the staircase. And all of the other storage classes are in between.

-> With lifecycle configurations, this staircase **only goes one way: down. Once you transition data down the staircase to a lower-cost storage class, you can't move objects back up.**

-> Lifecycle Configuration costs follow a similar staircase model. I categorize these costs in two ways: **minimum storage duration fees**, and **storage transition costs**. **Both of which increase as you move down the staircase.**

: **storage transition costs.** You get charged when you move data to other storage classes and this fee increases as you move down the staircase. For example, at the top of the staircase, you're charged **$0.01 for every 1,000 lifecycle transition requests when objects are moved from S3 Standard to the S3 Standard-IA storage class.** As you go down the staircase, all the way to **S3 Glacier Deep Archive**, this cost increases, and can be up to **$0.05 for every 1000 transition requests.**
-> if you need to transition millions of small objects to archival storage, that transition cost can be very high. To minimize this cost, you should **consider transitioning mostly large objects that need to be retained over long periods of time.** You can also consider **aggregating several small objects into one large object to save on this fee as well.**

: **minimum storage duration fees.** Most storage classes have a minimum storage duration that requires you to keep data in a storage class for a certain period of time before you delete, overwrite, or transition those objects. **These minimum storage duration periods increase as you go down the staircase as well.** For example,

S3 Standard and S3 Intelligent-Tiering have no minimum storage duration.

Infrequent access tiers like S3 Standard-IA and S3 One Zone - IA have a minimum storage duration of 30 days.
Archival storage tiers like S3 Glacier Instant Retrieval and S3 Glacier Flexible Retrieval have a minimum storage duration of 90 days
S3 Glacier Deep Archive has a minimum storage duration of 180 days.

-> So what happens if you delete or overwrite these objects before the minimum storage duration is reached? You get charged. For example, say you transition an object into S3 Glacier Deep Archive for 30 days, and then delete it. In this case, you will still be charged for the full 180 days of storage.

>>
The Amazon Elastic File System (EFS):
>>

-> **Amazon Simple Storage Service or S3 is an object storage solution:**
: Object storage stores **everything as a single object**, not in small chunks or blocks.
: With this type of storage, you upload a file and if the file changes to replace it, **the entire file will be replaced.**
: This type of storage is **best for situations where files are written once and then accessed many times.**
: It's **not optimal for situations that require both heavy read and write access at the same time.**
: So Amazon S3 is usually **used for storage of large files such as video files, images, static websites, and backup archives.**
: For example, **Netflix uses S3 for their data streaming service**. They upload large movie files once and then subscribers access and play the movies many, many times.

-> **Amazon Elastic Block Store or EBS is block-level storage:**
: Files are **not stored as single objects.**
: They're **stored in small chunks of blocks so that only the portion of the file that is changed will be updated**.
: This type of storage is **optimized for low latency access and when fast, concurrent read and write operations are needed.**
: EBS **provides persistent block storage volume**s for use with a single EC2 instance. i.e, even if you stop or terminate an EC2 instance that's using EBS, the data on the EBS volume remains intact.
: You should use this type of storage to store **operating system files, applications and other files you wish to obtain for use with your EC2 instance.**

-> Amazon Elastic File System (EFS) is considered file-level storage:
: It is also **optimized for low latency access**, but unlike EBS,
: it supports **access by multiple EC2 instances at once**.
: It appears to users like a file manager interface and **uses standard file system semantics such as locking files, renaming files, updating files and uses a hierarchy structure.**
: This type of storage allows you to **store files that are accessible to network resources.**

-> In traditional premises-based networks, users access files by browsing network resources that connect to a server, perhaps **via a mapped drive** that has been configured for them, and once they connect, they will see a tree view of available folders and files. This functionality is generally provided by various **local area network systems such as**
- file servers or
- storage area network, a SAN, or
- network-attached storage, a NAS.

-> EFS provides simple, scalable file storage for use with Amazon EC2 instances. Much like traditional file servers, or a SAN or a NAS, Amazon EFS provides **the ability for users to browse cloud network resources.**
-> EC2 instances can be figured to access Amazon EFS instances using **configured mount points** which can be created in multiple availability zones that attach to multiple EC2 instances.
-> So, much like your traditional land servers, **EC2 instances are connected to a network file system, Amazon EFS**. So from a user standpoint, the result is the same. The user accesses network resources just as they always have done except for now, it's done using cloud resources.
-> EFS is a **fully managed, highly available and durable service** that **allows you to create shared file systems that can easily scale to petabytes in size with low latency access.**
-> EFS has been designed to maintain a **high level of throughput in addition to low latency access response,** and these performance factors make EFS a desirable storage solution for a wide variety of workloads, and use cases and can meet the demands of tens, hundreds or even thousands of **EC2 instances concurrently.**
-> Being a managed service, **there is no need for you to provision any file servers to manage the storage elements or provide any maintenance of those servers**. This makes it a very simple option **to provide file-level storage within your environment.**
-> It uses standard operating system APIs, so any application that is designed to work with standard operating system APIs will work with EFS.

-> It supports both **NFS versions 4.1 and 4.0, and uses standard file system semantics such as strong consistency and file locking.**
-> It's replicated across availability zones in a single region making EFS a highly reliable storage service.
-> As the file system **can be accessed by multiple instances,** it makes it a very **good storage option for applications that scale across multiple instances allowing for parallel access of data.**
-> The EFS file system is also **regional,** and so any application deployments that span across multiple availability zones can all access the same file systems providing a level of high availability of your application storage layer.

: Storage classes and Performance options:

-> Different storage class options that EFS provides in addition to how you can alter and configure certain performance factors depending on your use case of EFS.
-> Amazon EFS offers two different storage classes, which each offer different levels of performance and cost:
Standard: The Standard storage class is the default storage used when using EFS. With the Standard storage class, you are only charged on the amount of storage space used per month.
Infrequent Access (IA): Infrequent Access is generally used if you're storing data on EFS that is rarely accessed. When using IA, you are charged for the amount of storage space used, which is cheaper than that compared to Standard. You are also charged for each read and write you make to the storage class. This helps to ensure that you only use this storage class for data that **is not accessed very frequently,** for example, data that might be required for auditing purposes or historical analysis.
-> Both storage classes are available in all regions where EFS is supported. And importantly, they both provide the same level of availability and durability.
-> For data management, within EFS, a similar feature exists known as **EFS lifecycle management.** When enabled, EFS will automatically move data between the Standard storage class and the IA storage class. This process occurs when a file has not been read or written to for a set period of days, which is configurable, and your options for this period range include 14, 30, 60, or 90 days.
-> EFS will move the data to the IA storage class to save on cost once that period has been met. However, as soon as that same file is accessed again, the timer is reset, and it is moved back to the Standard storage class. Again, if it has not been accessed for a further period, it will then be moved back to IA. Every time a file is accessed, its lifecycle management timer is reset. **The only exceptions to data not being moved to the IA storage class is for any files that are below 128K in size and any metadata of your files, which will all remain in the Standard storage class.**
-> If your EFS file system was created after February 13th, 2019, then the life cycle management feature can be switched on or off.
-> AWS are where the EFS can be used for a number of different use cases and workloads, and as such, each use case might require a change of performance from a **throughput, IOPS, and latency point of view**.

-> AWS has introduced two different **performance modes** that can be defined during the creation of your EFS file system:
General Purpose: General Purpose is a default performance mode and is typically used for most use cases. It offers an **all-round performance and low latency file operation,** and **there is a limitation of this mode allowing only up to 7,000 file system operations per second to your EFS file system.** When using the General Purpose mode of operations, EFS provides a CloudWatch metric percent I/O limit, which will allow you to view operations per second as a percentage of the top 7,000 limit. This allows you to make the decision to migrate and move to the Max I/O file system, should your operations be reaching that limit.
Max I/O: this mode offers **virtually unlimited amounts of throughput and IOPS**. If you have a huge scale architecture, where your EFS file system is likely to **be used by many thousands of EC2 instances concurrently**, and will exceed 7,000 operations per second, then you'll need to consider Max I/O. Your file operation latency will take a negative hit over that of General Purpose.

-> EFS also provides two different **throughput modes**, and throughput is **measured by the rate of mebibytes**. The two modes offered are:
Bursting Throughput : Data throughput patterns on file systems generally go through periods of relatively low activity with occasional spikes in burst usage. The default throughput available is capable of bursting to 100 mebibytes per second, however, with the standard storage class, this can burst to 100 mebibytes per second per tebibyte of storage used within the file system.
Provisioned Throughput : EFS provisions throughput capacity help to **manage the random activity of high peaks**.If you are finding that you're running out of burst credits too often, then you might need to consider using the Provisioned Throughput mode. Provisioned Throughput allows you to burst above your allocated allowance, which is based upon your file system size.

-> The amount of throughput scales as your file system grows.
-> The duration of throughput bursting is reflected by the size of the file system itself.
-> Every file system can reach its baseline throughput 100% of the time. By accumulating, getting credits, your file system can then burst above your baseline limit. The number of credits will dictate how long this throughput can be maintained for, and the number of burst credits for your file system can be viewed by monitoring the **CloudWatch metric of BurstCreditBalance.**

-> **Example**: So if your file system was relatively small but the use case for your file system required a high throughput rate, then the default bursting throughput options may not be able to process your request quick enough. In this instance, you would need to use provisioned throughput. However, this option does incur additional charges, and you'll pay additional costs for any bursting above the default option of bursting throughput.

: Mounting:

-> EFS offers two methods to **connect your Linux-based EC2 instance to your EFS file system**. Both use a process called **mounting** whereby you mount a target to the EFS file system on your instance.

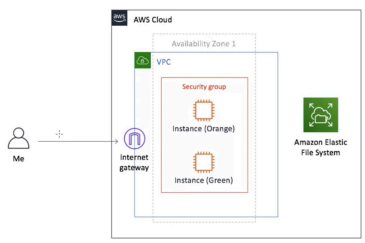
The original method available with EFS used the **standard Linux NFS client** to perform the mount.

The newer method is now the preferred option, and this uses the **EFS mount helper.**

-> **The EFS mount helper** is a utility that has to be installed on your EC2 instance. This utility has been designed to simplify the entire mount process by using predefined recommended mounting options that are commonly used within the NFS client. It also provides built-in login capabilities to help with any troubleshooting that might be required and are stored in the following location: /var/log/amazon/efs

-> EFS mount helper to automatically connect to EFS during the boot process, as well as by editing the /etc/fstab configuration file.

-> Before using the EFS mount helper to connect to your EFS file system from your EC2 instances, there are a couple of **prerequisites** required to be in place.



- You need to ensure that you have **created and configured your EFS file system, in addition to your EFS mount targets.**
- You must have an **EC2 instance running with the EFS mount helper installed,** and this instance will be used to connect to the EFS file system.
- The instance must also be in the **VPC and configured to use the Amazon DNS servers with DNS hostnames enabled.**
- You must have **a security group configured allowing the NFS file system NFS access to your Linux instance,**
- You must also be able to connect to your Linux instance.

-> configuration points related to the following:

EC2 security groups,
Mount targets,
Lifecycle management,
Throughput mode,
Performance mode,
Encryption.

Example: **Infrastructure-** just a VPC with a public subnet, with a couple of instances running in them with a security group associated to them, allowing me SSH access.

-> creating this new security group, allowing my instances to write to **EFS using the NFS protocol.** Needs to be associated with the mount points to allow the EC2 instances to write to the EFS file system.

-> There are three different steps to creating your file system.

Firstly, we need **to configure the file system access**. And we can see here that it says, "An Amazon EFS file system is accessed by EC2 instances running inside one of your VPCs. Instances connect to your file system using a network interface called a **mount target.** Each mount target has an IP address, which we (AWS) will assign automatically or you can simply specify your own."

Add the security group : So as per this security group, this mount target will allow inbound NFS traffic from the source specified in the security group, which were my EC2 instances. So that allows those EC2 instances to write to the EFS file system.

Secondly, we can **configure optional settings.** You can add **tags**, **Lifecycle management**, that means is any files that are not accessed for 30 days will be moved to the Infrequent Access storage class to save on cost. And then as soon as they are accessed again, then it will be moved back into the Standard storage class; two **throughput modes**, Bursting or Provisioned; Then we have our **performance mode,** General Purpose, or Max I/O. we **can enable encryption** at rest. EFS has selected this **default KMS master key,** the default master key that will be used by EFS.

Final step where we **can review and create our EFS file system**: we just review the options we've selected, make sure we have the **right availability zones selected, the security group**, and also **optional settings** as well.

Then mount our newly created file system to the directory that we just created. Both of the EFS directories on each EC2 instance is now associated to my EFS mount target. file is being stored on EFS and not locally on each of your two instances. So it's a shared location, and all we've done is simply created a new directory on each of those instances, and associated the

EFS file server with each of those directories.

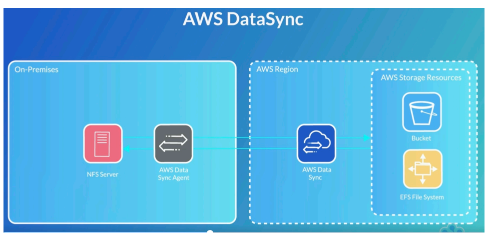
-> To initially create your EFS file system, you need to ensure that you have allow access for the following services:
elasticfilesystem:CreateFileSystem
elasticfilesystem:CreateMountTarget
ec2:DescribeSubnet
ec2:CreateNetworkInterface
ec2:DescribeNetworkInterfaces

Encryption at rest. This uses another AWS service, the key management service known as **KMS**, to manage your encryption keys. a KMS master key, a CMK, is required.
A customer master key is the main key type within KMS. This key can encrypt data of up to four kilobytes in size, however, it is typically used in relation to your data encryption keys. The CMK can generate, encrypt, and decrypt these data encryption keys, which are then used outside of the KMS service by other AWS services to perform encryption against your data, for example, EFS.

-> there are two types of customer master keys.
    Firstly, those which **are managed and created by you and I as customers of AWS**, which can either be **created using KMS**, or by importing key material from existing key management applications into a new CMK
    Secondly, those that are **managed and created by AWS themselves.**

-> The CMKs which are managed by AWS are used by other AWS services that have the ability to interact with KMS directly to perform an encryption against data, for example EFS, to perform its encryption at rest across its file systems.
-> These AWS managed keys can only be used by the corresponding AWS service that created them within the particular region, as **KMS is a regional service.**
-> These CMKs that are used by the services are generally created the first time you implement encryption using that particular service.

Encryption in transit. If you need to ensure your data remains secure between the EFS file system and your end client, then you need to implement encryption in transit. The encryption is enabled through the utilization of the TLS protocol, which is transport layer security, when you perform your mounting of your EFS file system. The best way to do this is to use the EFS mount helper. The command used to implement the use of TLS for in-transit encryption is as follows: sudo mount -t efs -o tls fs-12345678:/ /mnt/efs



-> This will ensure that the mount helper creates a **client stunnel process using TLS version 1.2.**

-> **stunnel** : is an open-source multi-platform application used to provide a universal TLS/SSL tunneling service. Stunnel can be used to provide secure encrypted connections for clients or servers that do not speak TLS or SSL natively.' This stunnel process is used to listen out for any traffic, using NFS, which it then redirects to the client stunnel process.

>>**>**
AWS DataSync :
>>>

-> **AWS DataSync.** This service is specifically **designed to help you securely move and migrate and synchronize data for your existing on-premises site into AWS Storage Services such as Amazon EFS or Amazon S3 with simplicity and ease.**
-> AWS DataSync, it's a service that **allows you to easily and securely transfer data**
    : from your Snowcone or your on-premise data center, to AWS storage services.
: **from your on-premise data center to AWS storage services.**
-> It can **also be used to manage data transfer between 2 different AWS storage services** too,
-> it's a great **service to help you migrate, manage, replace and move data between different storage locations.**
-> supports the ability to work with data stored on
    Network File Systems shares,
    Server Message Block shares, and
    any self-managed object storage,
-> in addition to the following AWS services:
    Amazon S3
    Amazon Elastic File System
    Amazon FSx for Windows File Server
    AWS Snowcone

-> The data transfer can either be accomplished over a direct connect link or over the internet.
-> When performing data transfer operations, DataSync **support AWS VPC Endpoints** so
- its able to utilise the high bandwidth,
- low latency AWS network to it's advantage,
- this helps to both simplify the management of the request and automate your data transfer across secure infrastructure.

-> AWS Data Sync comes with its **own purpose-built data transfer network protocol** in addition **to a parallel and multithreaded architecture to rapidly perform data transfer**, i.e each DataSync task has the **potential of utilizing 10 Gbps** over a network link between your own on-prem data center and your AWS environment.
-> To sync source files from your on-premises environment, you must download the DataSync agent as a **VMware ESXi** to your site. The agent is configured with the source and destination target and associated with your AWS account, and logically sits in between your on-premise file system and your EFS file system.

->AWS DataSync provides 2 levels that provide end-to-end security,
: **encryption :**
- supports encryption **in transit** is implemented by encrypting the data using the Transport Layer Security (TLS) protocol.
- supports encryption **at rest** mechanisms that **EFS and FSx for Windows service offers,** also the **default encryption at rest option for Amazon S3.**
: **data validation:**
- ensures that your data arrives at its destination in one piece, exactly as it was when it left the source ensuring that it wasn't compromised or damaged in any way during its transit.
- check helps to validate the consistency of your data that was written to the AWS storage service, and that its a perfect match from when it left its source location.

-> DataSync is also very useful if you want to **transfer files between EFS file systems either within the same AWS account or cross-account and owned by a third-party.**
-> To help with the management and implementation of this transfer, AWS has created an **AWS DataSync in-cloud** quick start and scheduler.

-> Includes:
- You can migrate an **NFS file system from Amazon EC2 to Amazon EFS within the same AWS region**.
- **Replicate an NFS file system from Amazon EC2 in one AWS region to an Amazon EFS file system in a different AWS region for disaster recovery.**
- You can **migrate an Amazon EFS file system from EFS standard with no lifecycle management to an EFS file system with lifecycle management enabled.**
- File systems with lifecycle management enabled will automatically move to a lower-cost Infrequent Access storage class based on a predefined lifecycle policy.
- You can migrate an **Amazon EFS file system from one performance mode** to another **performance mode within the same AWS region**,
- **replicate an Amazon EFS file system from one AWS region to another Amazon EFS file system in a different AWS region for disaster recovery.**

-> AWS DataSync uses a **flat pricing strategy based on a per-gigabyte of data transferred,** this makes it easy to predict avoiding any unexpected costs.

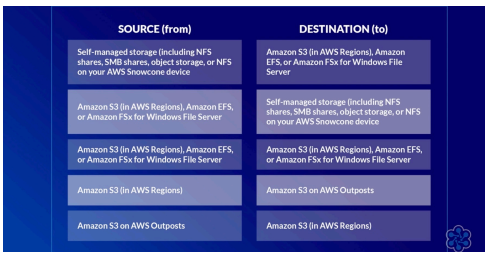
AWS DataSync use case:
1. is archiving data into cold storage:
By utilising AWS DataSync you **can schedule a task to migrate this data from your own data centre into one of the Glacier storage services (**Amazon S3 Glacier storage classes, (Glacier Deep Archive (cheapest storage solution provided by AWS)) primarily used for archiving rarely accessed data**)**. This then **ensures the durability of the data at a very low cost without the worry of maintaining that data yourself on-premises.** This would also likely **allow you to remove any old and perhaps legacy storage solution that you were using to store this data yourself.**

2. to the need to implement steady and active data migrations to Amazon S3, Amazon FSx for Windows File Server or EFS on **a regular basis.**
Depending on your workload, you might want to consider this option for **potential backup purposes or DR**, for example you might migrate data to EFS for a standby file system in the cloud should your primary on-site file system suffer an outage.

3. utilising a **hybrid cloud solution**, utilising services and solutions both on premises and in AWS.
Need to harness the power and speed that many of the AWS services have, this is especially true if you are working with **Machine learning or trying to process data sets in a short period of time.** You could migrate the data into AWS for this additional processing and analysis and then migrate data back with any results into your Data Centre when your operations

have completed.

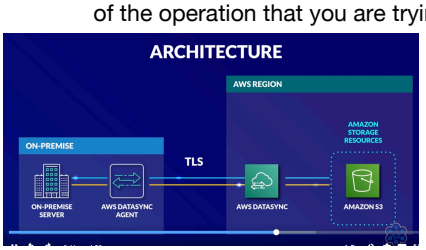
NOTE: when using DataSync **it will not copy and configuration relating to the source storage option,** for example if you were to copy from one S3 bucket to another S3 bucket, then it would **only move the data,** it would **not copy any bucket-level settings or permissions.**

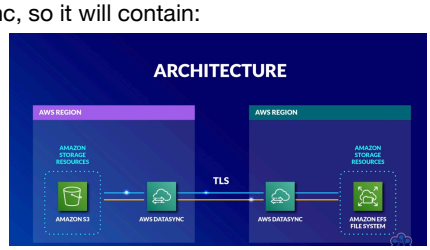


AWS DataSync Architecture (operational level) :
-> architecture of the service is put together and the different components involved to carry out a DataSync task.

Case 1: When transferring data from your own managed storage environment to AWS
-> we have our own self-managed storage solution on-premises and we need to use AWS DataSync to move this data into Amazon S3
- we need to **configure an Agent, a Location and a Task.**

- **The Agent** will be used on the customer side, so it sits outside of AWS, and it's just a virtual machine supported by VMware ESXi, KVM or Microsoft Hyper-V hypervisors, so it should be compatible with your existing infrastructure.
- The agent itself is used to both read and write data to your own storage solution and can be generated and configured from within the AWS Management dashboard which can then be downloaded.
- **The location** identifies the endpoint of a DataSync task. So everytime you create a DataSync task you will need to specify the source location and the destination location; can be configured from within the AWS Management Console, dictating where you want to move data from and to.
  - You can create locations for:
    Network File Systems (NFS)
    Server Message Blocks (SMB)
    Self-managed object storage
    Amazon EFS
    Amazon FSx for Windows File Server
    Amazon S3
- **The task** : essentially outlines exactly what will happen during the data transfer process. The task contains the details of the operation that you are trying to carry out and perform with DataSync, so it will contain:





  - the locations that were created and specified for both the source and destination,
  - the configuration and conditions of how the data transfer will take place.
  - DataSync tasks will only copy your storage data, it doesn't include any file systems permissions or settings.
- For example:
  - configure the type integrity and data verification checks to take place
  - to transfer all data in the source location or If you only want to transfer specific files from the source, then you can apply pattern filters enabling you to restrict which files to include or exclude from the transfer in the source location.
  - data that has changed since the last task was performed
  - to overwrite or delete files
  - logging details which integrate with Amazon CloudWatch Logs to help you identify any failures or errors.

Architecture:

Case 1: When transferring data from your own managed storage environment to AWS
-> We have the on-premises server holding our storage data with the AWS DataSync agent installed as a virtual machine.
-> Two locations would have been created, with the source pointing to the on-premises server and the destination to an S3 bucket.
-> The task would then be configured to transfer the data conforming to the setting configured within the task,
-> when it runs AWS DataSync will transfer the data using encryption-in-transit over TLS to the Amazon S3 destination bucket.
-> All logging information would then be stored in Amazon CloudWatch if configured to do so.

Case 2: When transferring data between 2 different AWS storage services, such as Amazon S3 to Amazon EFS
-> in this process **we do not use the Agent**, i.e we don't need to use the DataSync agent.
-> **Location:** to create 2 locations, a source location for Amazon S3 and a destination location for EFS.

-> **Task:** we will also have to create a Task. So the process remains very similar to that of when transferring data from on-premises, however, we don't need to use the DataSync agent.
>>>**>**

EC2 Instance Level Storage:

>>>

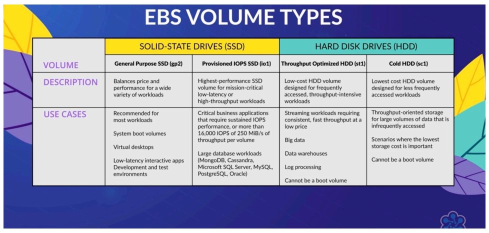
EC2 Instance Level Storage (an instance store volume):

-> the volumes physically reside on the **same host that provides your EC2 instance itself,** acting as local disc drives, allowing you to store data locally to that instance.
-> instance store volumes provide **ephemeral storage** for you EC2 instances. Ephemeral storage means that the **block level storage that it provides offers no means of persistency.** Any data stored on these volumes is **considered temporary**. With this in mind, it is not recommended to store critical or valuable data on these ephemeral instance store volumes, as it could be lost, should an event occur.
-> If your **instance is either stopped or terminated,** then any data that you have stored on that instance store volume associated with this instance will be deleted without any means of data recovery.
-> if your **instance was simply rebooted, your data would remain intact**.
-> Although, you can control when your instances are stopped or terminated, giving you the opportunity to either back-up the data or move it to another persistent volume store, such as the elastic block store service. **Sometimes this control is not always possible**.
Example: Let's consider you had critical data stored on an ephemeral instance store volume and then the underlying host that provided your EC2 instance and storage failed. You had no warning that this failure was going to occur, and as a result of this failure, the instance was stopped or terminated. Now all of your data on these volumes is lost.
-> When a **stop and start**, or **termination** occurs, **all the blocks on the storage volume are reset, essentially wiping data.**
-> **Not all instance types support instance store volumes.**
-> From a security stance, instance store volumes **don't offer any additional security features.** At most, This can be **IAM policies dictating which instances can and can't be launched, and what action you can perform on the EC2 instance, itself.**
-> From a cost perspective, **the storage used is included in the price of the EC2 instance**. So, you **don't have an additional spend on storage cost.**
-> The I/O speed on these volumes can far exceed those provided by the alternative instance block storage, EBS
Example: When using **store optimized instance families,** such as the **I3 Family**, it's potentially possible to reach 3.3 million random read IOPS, and 1.4 million write IOPS. With speeds like this, it makes it ideal to handle the high demands of no SQL databases. However, **any persistent data required would need to be replicate or copied to a persistent data store in this scenario.**
-> Instance store volumes are generally **used for data that is frequently changing;** that doesn't need to be retained, as such, they are **great to be used as a cache or buffer.**
-> They are also commonly **used for service within a load balancing group, where data is replicated across the fleet such as a web server pool.**
>>>**>**

Amazon Elastic Block Store service (EBS) :

>>>

-> If you need to use block level storage and want a quick and easy method to maintain persistency, then there is another block level service that is recommended. This being the **elastic block store service.**
-> Amazon Elastic Block Store service, known as EBS, which provides storage to your EC2 instances **via EBS volumes**, which offer different benefits to that of instance store volumes used with some EC2 instances.
-> EBS provides **persistent and durable block level storage**.
-> EBS volumes offer far **more flexibility with regards to managing the data when compared to data stored on instance store volumes.**
-> EBS volumes can be attached to your EC2 instances, and are primarily **used for data that is rapidly changing that might require a specific Input/Output operations Per Second rate (IOPS).**
-> EBS volumes are independent of the EC2 instance, meaning that they exist as two different resources. **They are logically attached to the instance** instead of directly attached like instance store volumes. Due to the EBS ability to enforce persistence of data, it doesn't matter if your instances are intentionally or unintentionally stopped, restarted, or even terminated, the data will remain intact when configured to do so.
-> From a connectivity perspective, **only a single EBS volume can only ever be attached to a single EC2 instance**. However, **multiple EBS volumes can be attached to a single instance.**



-> snapshots: EBS also offers the **ability to provide point in time backups of the entire volume as and when you need to**. These backups are known as **snapshots** and you can manually invoke a snapshot of your volume at any time, or use Amazon CloudWatch events to perform an automated schedule of backups to be taken at a specific date or time that can be recurring.
-> The snapshots themselves are then stored on Amazon S3 and so are **very durable and reliable.**
-> They **are also incremental,** meaning that each snapshot will **only copy**

data that has changed since the previous snapshot was taken.
-> Once you have a snapshot of an EBS volume, you can then create a new volume from that snapshot. So, if for any reason you lost access to your EBS volume through or incident or disaster, you **can recreate the data volume from an existing snapshot and then attach that volume to a new EC2 instance.**
-> To **add additional flexibility and resilience,** it is **possible to copy a snapshot from one region to another.**
-> Every write to a EBS volume is **replicated multiple times within the same availability zone of your region** to help prevent the complete loss of data. This means that your **EBS volume itself is only available in a single availability zone.** As a result, should your availability zone fail, you will lose access to your EBS volume. Should this occur, you can simply **recreate the volume from your previous snapshot and attach it to another instance in another availability zone.**
-> There are two types of EBS volumes available.

SSD backed storage (solid state drive): SSD backed storage is better suited for scenarios that work with smaller blocks. Such as **databases using transactional workloads.** Or often as **boot volumes for your EC2 instances.**

HDD backed storage (hard disk drive): HDD backed volumes are designed for better **workloads that require a higher rate of throughput**, such as **processing big data and logging information**. So, essentially working with larger blocks of data.

-> These volume types can be broken down even further.

General Purpose SSD (gp2)

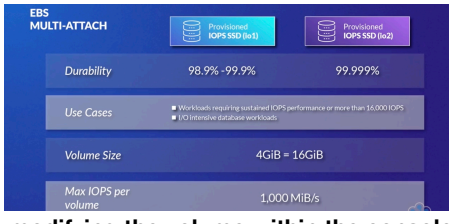
Provisioned IOPS SSD (io2): they deliver enhanced predictable performance for applications requiring **I/O intensive workloads**. When working with these volumes you also have the ability to specify at IOPS rate during the creation of a new EBS volume, and when the volume is attached to an EBS-optimized instance, EBS will deliver the IOPS defined and required within 10%, 99.9% of the time throughout the year.

Throughput Optimized HDD (st1): The throughput optimized HDD volumes are designed for **frequently accessed data** and are ideally suited to work well with **large data sets requiring throughput-intensive workloads,** such as **data streaming, big data, and log processing.** These volumes will deliver the expected throughput 99% of the time over a given year, and an important point to make is that these volumes **can't be used as boot volumes for your instances.**

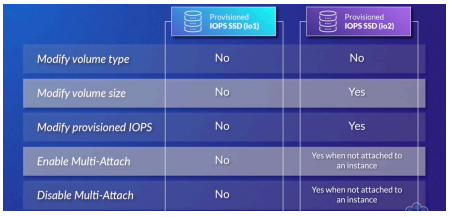
Cold HDD (sc1): The cold HDD volumes offer the **lowest cost** compared to all other EBS volumes types. They are suited for workloads that are **large in size and accessed infrequently.** They will deliver the expected throughput 99% of the time over a given year, and again, it is **not possible to use these as boot volumes for your EC2 instances.**

-> **EBS data encryption:** Ability to enhance the security of your data, both at rest and when in transit,
-> This is especially useful when you have sensitive data, such as personally identifiable information, stored in your EBS volume. And in this case, you may be required to have some form of encryption from a regulatory or governance perspective.
-> EBS offers a very simple encryption mechanism. Simple in the fact that you don't have to worry about managing the data keys to perform the encryption process yourself. It's all **managed and implemented by EBS.**
-> The encryption process uses the **AES-256 encryption algorithm** and **provides its encryption process by interacting with another AWS service, the key management service, known as KMS**.
-> KMS uses **customer master keys, CMKs,** enabling the encryption of data across a range of AWS services, such as EBS in this instance.
-> Any snapshot taken from an encrypted volume will also be encrypted, and also any volume created from this encrypted snapshot will also be encrypted.
-> encryption option is only available on selected instance types.
-> EBS encryption can create a **default region setting** that ensures that all EBS volumes created will be encrypted by default.

-> As EBS volumes are separate to EC2 instances, you can create an EBS volume in a couple of different ways from within the management console. **During the creation of a new instance and attach it at the time of launch**, or **from within the EC2 dashboard of the AWS management console as a standalone volume ready to be attached to an instance when required.**



-> EBS volumes **can only be attached to EC2 instances that exist within the same availability zone.**
-> EBS volumes are **able to resize them elastically**. This can be achieved by **modifying the volume within the console** or **via the AWS CLI** or **perform the same resize of the volume by creating a snapshot of your existing volume, and then creating a new volume from that snapshot with an increased capacity size.**



-> if you only needed temporary storage or multi-instance storage access, then EBS is not recommended, as **EBS volumes can only be accessed by one instance at a time.**
-> if you needed very high durability and availability of data storage, then you would be better suited to use Amazon S3 or EFS, the elastic file system.

: Multi-Attach:
-> **Amazon Elastic Block store** is used to provide **persistent block level storage** for your EC2 instances which can be

detached and reattached to another instance running in the **same availability zone**. For most volume types it's only possible to have **each EBS volume attached to a single EC2 instance at any one tim**e, however, **with EBS Multi-Attach you can allow multiple instances to access the same volume.**

-> this feature is very dependent on the **type of volume** and also the **type of instance**.
-> it's only possible to use Multi-Attach with 2 different volume:
Provisioned IOPS SSD (io1)
Provisioned IOPS SSD (io2)

-> Provisioned IOPS SSD volumes are typically used for scenarios that **require high performance with low latency or high throughput,** so they are more of a specialized volume type designed to help you deliver those missions **critical workload**s in your business.
-> these maximum performance ratings can only be performed on EC2 instances that are **based on the Nitro system.**
-> When it comes to performance, the **aggregate performance of each of the connected instances can't exceed the maximum IOPS for the shared volume.** So if you set your Max IOPS of the volume at 32,000, and your instances could reach a max of 30,000 IOPS, then an individual instance could utilize its own maximum allowance, however, if 2 instances were trying to perform operations at the same time they would have to share the total available IOPS of the volume and would not be able to exceed 32,000.

-> the **io2 volume provides additional management support** over that of the io1 volume type, but **neither allows you to modify the size of the volume once it has been created.**

-> using **Nitro instances is a requirement of using Multi-Attach**, and so it's **only available for instances that are Nitro-based.**
-> **Nitro**, it basically refers to the **underlying virtualization platform of the EC2 instance.** It's designed to be **faster, cheaper** and offers additional benefits such as **enhanced security** features.
-> The Nitro system also allows the introduction of **bare metal instances** which means **you can run an EC2 instance without any hypervisor**, or we as the customer, can use our own hypervisors.

operational components:
-> when you **enable multi-attach volumes they do not support I/O fencing.**
-> you must create your volume in the same availability zone as the instance you want to attach it to, as much like normal EBS volumes you **can only attach them to resources that exist in the same availability zone.**
-> When connecting the volume to your **Linux instances then you have a limit of 16 Nitro instances being attached to the same volume, and this rule applies for both io1 and io2 volumes**. If you are running **window instances then Multi-Attach enabled volumes can be associated with the instance,** however, the Windows OS will not recognize that it is a shared volume being accessed by multiple instances. As a result, this can mean **data inconsistencies** and so it should be avoided and **used for Linux instances** instead.
-> **AWS does not recommend using standard file systems like XFS or EXT4** as it can lead to **data loss** as these file systems are not designed to be accessed by multiple instances at once. Instead, you are recommended to **use a clustered file system.** An example of this could be **GFS2** which will safely manage multi-instance access to the shared storage volume.
-> many normal EBS features such as **encryption**, the way in which you **attach a volume to an instance** remains the same, and also you still have the **delete on termination option available with your EC2 instance**s.
-> The settings would be **applied based on the last EC2 instance that was connected to the volume**.

Amazon Data Lifecycle Manager:
what are **EBS volumes**?
Amazon Elastic Block Store isn't easy to use scalable, high performance block storage service designed for Amazon Elastic Compute Cloud or Amazon EC2.
What are **snapshots**?
Snapshots are a convenient way to back up your block level data regardless of where it resides.

what is **Amazon DLM**?
DLM is **Amazon Data Lifecycle Manager** that **allows you to automate the creation, deletion, and retention of EBS snapshots, and EBS backed AMI's.** Amazon Data Lifecycle Manager is provided to you at no extra cost.

-> AWS Cloud, one of the key cloud optimization tools that AWS has provided was tags, make sure you always tag your instances. AMI's, Volumes, and Snapshots. **If you aren't tagging your resources, Amazon DLM won't work, automation won't work, and scheduling won't work.**
-> DLM can also be used in conjunction with other AWS services. For example, Amazon CloudWatch Events. AWS CloudTrail to provide a complete backup solution for your EC2 instances.

>>
Amazon FSx :

>>>

-> Amazon FSx is another storage service that focuses on file systems, much like EFS. However, FSx comes in 2 forms:

Amazon FSx for Windows File Server:

- Provides a fully managed native Microsoft Windows file system on AWS.
- easily move and migrate your windows-based workloads requiring file storage.
- solution is built upon Windows Server
- operates as shared file storage
- it uses SSD storage for enhanced performance and throughput providing sub-millisecond latencies
- It has full support for:
  SMB protocol,
  Windows NTFS,
  Active Directory (AD) integration, and
  Distributed File System (DFS)
- has 3 price points:

Capacity: There are no setup fees for the use of this service, however, you do pay for the amount of storage capacity that you consume. This is priced on the average storage provisioned per month and uses the metric of gigabyte-months and offers varied pricing between a single or multi-AZ deployment.

Throughput: a cost of for the amount of throughput that you configure for your file systems, this metric is based upon MBps-months. Again, cost variations exist between single and multi-AZ deployment.

Backups: backup storage is also charged based on the average metric of gigabyte-months, meaning the average amount of capacity you have used in the month.

- data deduplication: FSx will automatically store duplicate files or portions of files a single time, this helps you save on your storage capacity costs and remains invisible to the user that the data has been deduplicated.
- Data deduplication doesn't add any additional costs and to help save up to 80% of storage costs.

Amazon FSx for Lustre:

- fully managed file system designed for compute-intensive workloads,
  for example, Machine Learning and high-performance computing.
- It has the ability to process massive data sets.
- Performance can run up to hundreds of GB per second of throughput, millions of IOPS, and sub-millisecond latencies.
- It has integration with Amazon S3
- supports and supports cloud-bursting workloads from on-premises over Direct Connect and VPN connections.

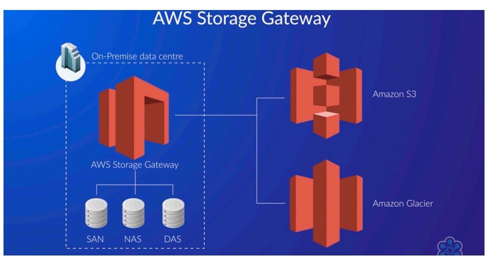
-> any data transfer costs when using multi-AZ is included in the pricing you see for the multi-AZ deployment.
->Amazon FSx performs incremental backups (either manual or automatic) of your file systems, it optimizes your storage costs as only the changes since the last backup are saved.
-> Much like your storage capacity, backup storage is also charged based on the average metric of gigabyte-months, meaning

>>>

Amazon AWS Storage Gateway :

>>>

-> AWS Storage Gateway allows you to **provide a gateway between your own data center's storage systems such as your SAN, NAS or DAS and Amazon S3 and Glacier.**



Storage Pricing	
File storage in S3	Stored and billed as Amazon S3 objects
Request Pricing	
Data written to AWS storage by your gateway	$0.01 per GB *
File storage in S3	Billed as Amazon S3 requests

-> The Storage Gateway itself can either be installed as software or physical hardware appliance that can be stored within your own data center which allows integration between your on-premise storage and that of AWS. This connectivity can allow you to scale your storage requirements both securely and cost-efficiently.

-> Storage Gateway offers different configurations and options allowing you to use the service to fit your needs. It offers file, volume and tape gateway configurations which you can use to help with your DR and data backup solutions, and each of these come with different price points.

-> **File Gateways**.

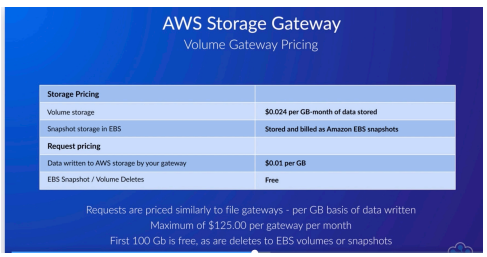
- File gateways allow you to securely **store your files as objects within S3.**
- This connectivity **acts as a type of file share allowing you to mount or map drives to an S3 bucket** as if the share was held locally on your own corporate network.
- In addition to this, **a local on-premise cache is also provisioned** for

accessing your most recently accessed files to optimize latency which also helps to reduce egress traffic costs.

- As this service integrates largely with Amazon S3, much of the **pricing is based upon S3 price points.**
- As the storage used for this type of Gateway **solely relies on S3, the actual storage costs are as per the Amazon S3 storage class price at the time of use**

- When we look at the request pricing **there is a small cost per GB associated to data written to S3 by the Storage Gateway, up to a maximum of $125.00 per gateway per month**. **Also, the first 100 GB is free**.
- All other request types again are as per Amazon S3's pricing.

-> **Volume Gateways.**



- Volume Gateways can be figured in one of two different ways:

Stored volume gateways: Stored volume gateways are **often used as a way to backup your local storage volumes to Amazon S3** as EBS snapshots whilst ensuring your entire data library also remains locally on-premise for very low latency data access.

Volumes created and configured within the storage gateway are **backed by Amazon S3** and **are mounted as iSCSI devices** that your applications can then communicate with.

Stored volume gateways create EBS snapshots, which are then stored on S3, however, they are **billed as Amazon EBS snapshots.**

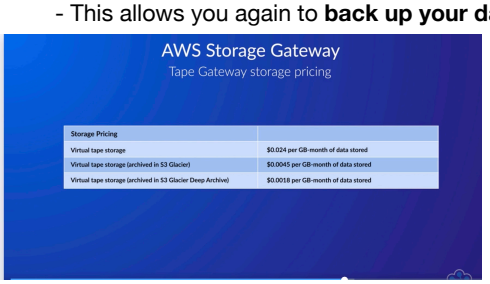
Cached volume gateways: With Cached volume gateways, **the primary data storage is actually on Amazon S3 rather than your own local storage solution.** However cache volume gateways **utilize your local data storage as a buffer and the cache for recently accessed data to help maintain low latency,** hence the name, Cache Volumes.

The cached volumes, however, are **charged on a per GB-month metric of data stored.**

- Any requests are priced similarly to File gateways in that they are **billed on a per GB basis of data written by the Gateway, up to a maximum of $125.00 per gateway per month.** Also**, the first 100 GB is free,** in addition to **any deletes to EBS volumes or snapshots also remain free.**

-> **Tape Gateways(Gateway VTL.: Virtual Tape Library):**

- This allows you again to **back up your data to S3 from your own corporate data center in addition to being able to leverage the storage classes within Glacier for data archiving for a far lower cost than S3.** Virtual Tape Library is essentially a **cloud-based tape backup solution replacing physical components with virtual ones.**



- These **are a virtual equivalent to a physical backup tape cartridge** and **any data stored on the virtual tapes are backed by AWS S3/Glacier and appear in the virtual tape library.** A Virtual Tape Library, VTL, as you may have guessed are virtual equivalents to a tape library that contain your virtual tapes.
- the pricing is split across storage and request pricing: There are 3 different options **for Storage pricing of Tape Gateways:**

S3:
S3 Glacier, and
S3 Glacier Deep Archive.

- All of which are **charged at per GB-month of data stored.**

Generally, if you are using Tape Gateways **you are looking to take advantage of the very low price points of Glacier and Deep Archive which offer significant savings**

- **Request pricing** also offers a range of different cost metrics **depending on the type of action and storage class used.**
- there is a **small cost per GB associated to data written to S3 by the Storage Gateway, up to a maximum of $125.00 per gateway per month.** Also, **the first 100 GB is free.**
- **For any virtual tape retrieval requests that are being stored on S3 Glacier classes, you will also pay a per GB cost, with Deep Archive providing a more expensive retrieval rate.**
- Any request that results in your **moving your virtual tapes between your S3 Glacier storage class and S3 Glacier Deep Archive you will pay a fee per GB of data moved.**
- If you do **select the Glacier storage classes and you delete your virtual tapes, within a set time period (90 days for S3 Glacier and 180 days for Deep Archive), then you will be charged a prorated charge per month per GB.**

AWS Storage Gateway uses a variety of storage options, **from Amazon S3**, **EBS Snapshots, Amazon S3 Glacier and S3 Glacier Deep Archive**, and the cost of each is dependent on the type of gateway required which will be dictated by your use case.

Pricing:

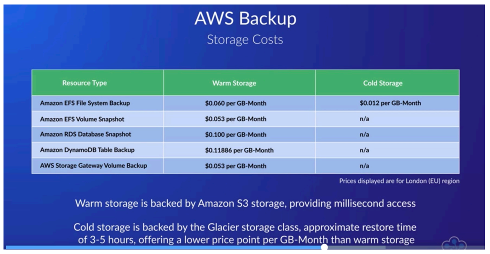
- File gateways pricing is very simple and essentially follows the pricing metrics of Amazon S3, apart from a per-GB request

for data writes.
- Volume gateways again offer a simple pricing structure but uses the per-GB metric for volumes in addition to any snapshots of the volumes priced at EBS snapshot costings.
- Tape gateways offer additional complexity due to the range of storage classes that it can use. So when using Tape gateways, understand the retrieval times for your data based on the S3 Glacier and Deep Archive classes as you might be able to save a considerable amount when using these classes depending on the criticality of the data you might need to retrieve.

>>**>**

Amazon AWS Backup :

>>



Resource Type	Warm Storage	Cold Storage
Amazon EFS File System Backup	$0.060 per GB-Month	$0.012 per GB-Month
Amazon EFS Volume Snapshot	$0.053 per GB-Month	n/a
Amazon RDS Database Snapshot	$0.100 per GB-Month	n/a
Amazon DynamoDB Table Backup	$0.11886 per GB-Month	n/a
AWS Storage Gateway Volume Backup	$0.053 per GB-Month	n/a

-> This is a solution to help you **manage and implement backups across a number of different supported AWS services.** -> It is also possible to automate the backup of data from on-premises thanks to its connectivity with the AWS Storage Gateway service.
-> AWS Backup **acts as a central hub to manage backups across your environment, across multiple regions, centralizing management and providing full auditability in addition to assisting with specific compliance controls.** -> Having a managed service monitor and control your backups **allows for all logging to be consolidated in a single place,** in addition to seeing **the status of completed backups and perform and restores required**.
-> **The service itself uses backup features from existing services,** so for example, if you were to manage your EBS backups, AWS Backup would manage these through the EBS Snapshot feature as a way of performing the backup.
-> When using AWS Backup you will need to create backup policies or backup plans. These simply determine the exact requirements that you need for your backups and contain information such as:

A backup schedule
Backup window
Lifecycle rules, such as the transition of data to cold storage after a set period
A backup vault, which is where your backups are stored and encrypted through the use of KMS encryption keys
Regional copies
Tags

-> From a cost perspective, the only real optimization available is when you are using services that support both warm and cold storage, where data is transitioned between the two via lifecycle rules which are configured within the backup plan.
-> **Warm storage** is backed by Amazon S3 storage providing millisecond access time.
-> **Cold storage** is as expected backed by the Glacier storage class, with an approximate restore time of 3-5 hours offering a lower price point per GB-month than warm storage. however, the retrieval time is a lot longer.

-> For backup storage when using AWS Backup, **all charges use the metric GB-month,** and depending on the resource type used, will depend on how much AWS Backup charges per GB-month.
-> However, with backup, comes the inevitable restore, and here there is also a cost implication.

>>>**>**

Amazon AWS Snow Family :

>>>

-> The snow family consists of a **range of physical hardware devices that are all designed to enable you to transfer data into AWS from the edge or beyond the Cloud, such as your Data Center, but they can also be used to transfer data out of AWS too, for example, from Amazon S3 back to your Data Centre.**

	AWS SNOWCONE	AWS SNOWBALL COMPUTE OPTIMIZED	AWS SNOWBALL COMPUTE OPTIMIZED WITH GPU	AWS SNOWBALL STORAGE OPTIMIZED	SNOWMOBILE
HEIGHT	3.25"	19.75"	19.75"	19.75"	9.6'
WIDTH	5.85"	12.66"	12.66"	12.66"	8'
LENGTH	8.94"	21.52"	21.52"	21.52"	45'
WEIGHT	2.1kg	21.3 kg	21.3kg	21.3kg	68,000 lbs
vCPU	2	52	52	24	N/A
MEMORY	4GB	208GB	208GB	32GB	N/A
STORAGE HDD	8TB	39.5TB	39.5TB	80TB	100PB
STORAGE SSD	N/A	7.68	7.68	N/A	N/A
GPU	N/A	N/A	NVIDIA Tesla V100 GPU	N/A	N/A

-> The snow family is different, instead, you will be sent a **piece of hardware packed with storage and compute capabilities to perform the required data transfer outside of AWS**, and when complete, the device is then sent back to AWS for processing and the data uploaded to Amazon S3.

:devices packing other options:
-> You can perform data transfers from as little as a few terabytes using an **AWS snowcone** all the way up to a staggering 100 petabytes using a single **AWS snowmobile**
-> Migrating and transferring data at high magnitude,
-> storage capacity for data transfer,
-> Battery packs

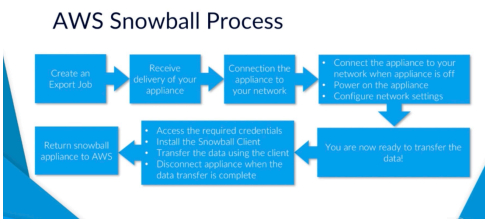
-> with compute power, allowing you to run usable EC2 instances that have been designed for the snow family enabling your applications to run operations in often remote and difficult to reach environments, even without having a data center in sight, and when working with a lack of persistent networking connectivity or power.
-> The enablement of running EC2 instances makes it possible to use these devices at the **edge to process and analyze data much closer to the source.**

-> **Snowcone** is the smallest followed by the **Snowball** and finally the **Snowmobile**.

: **The AWS Snowcone** is the
- smallest of the snow family, this has been designed to be
- lightweight,
- easily portable,
- use the device pretty much anywhere and under any conditions due to the ruggedness of the casing,
- run on battery should a persistent mains connection not be available. - It can easily fit into a standard backpack,
- can be attached to a drone emphasising its **portability and versatility**.
- **Example use case:**

1. Packed with 8TB of storage and an EC2 instance, this device is perfect for taking your computing needs way beyond the cloud and your Data Centre allowing you to capture, process, and analyze data, perhaps via IoT sensors, which can then be shipped back to AWS for data transfer, or you could even use AWS DataSync to transfer the data on-line over your traditional network connectivity.
2. **Needed a portable** device that you could easily carry to difficult to reach locations and situations
3. Only needed a **maximum of 8TB** storage
4. If you needed the ability to **perform on-line data transfer using AWS DataSync,** preventing you the need to send the Snowcone back to AWS for an off-line data transfer
4. If you didn't have a consistent power support and you **needed the support of a battery pack**
5. Snowcones only have **network port speeds of 10Gbit**

: **The AWS Snowball**



- it's bigger in size
- it contains a greater amount of storage and compute power.
- it's primarily used for large scale data transfer operations, up to 80TB at a time, both in and out of AWS.
- The devices themselves can be rack mounted in your data centre, and if need be clustered in groups of 5-10 devices.
- they can't be powered by battery expansion packs,
- they are not as portable.

: **The Storage optimized snowball** is targeted **for data migrations and transfers with its storage being compatible with both S3 object storage and EBS volumes.**

: **The Compute optimized snowball** is a great option if you need to handle compute intensive edge computing workloads in disconnected environments. From a storage perspective, it also comes

: The Compute Optimized with GPU option is used to accelerate AI, HPC, and graphics, which is great when working with video analysis and graphic intensive use cases.

- **Example use case:**

1. Didn't need to provide mobility to the snow device and it **could remain in one location for a set period of time**
2. Needed to **transfer data of up to 80TB** of usable HDD storage from a single snowball, it easily allows you to provide a means of **securely storing and transferring a large amount of data into AWS**,
3. can **run multiple snowballs in parallel allowing you to transfer petabytes of data if required.**
4. Needed the **ability to run enhanced graphics processing by using the Compute Optimized with GPU option**
5. Had a requirement to **transfer data using S3 API's**
6. Required the **use of usable SSD Storage**
7. Needed to **optimized network ports** that could reach **speeds of up to 100Gbit**,
8. Needed to **cluster** your snowballs.

- Clustering **allows you to order between 5-10 snowball devices, acting as a single pool of resources.**
- This allows you **to gain a larger storage capacity,** and also **enhance the level of durability of the data should a snowball fail.**
- **Clustering is only an option if you are looking to simply perform local compute and storage workloads without transferring any data back to AWS.**

9. Needed to **rack mount your devices** to implement temporary installations of both compute and storage
10. Required the snow device to be **HIPAA compliant**

- **Example use case:** Both **Snowcone** and **Snowball**

1. provide a level of portable edge computing allowing you to collect data from wireless sensors or networked resources.
Example: in locations such as industrial warehouses or manufacturing plants, where you might need to collect environmental metric data. By collecting and gathering data it can then be transferred to AWS offline, or if using the snowcone

it can be transferred on-line using AWS DataSync, which can then be analyzed at scale using other AWS services.

2. Being of rugged design and portable it is able to withstand the harshest of environments,

- the snowcone can operate in conditions of -32ºC/-25.6ºF to 63ºC/145.4ºF.
- It's all windproof, dustproof and water-resistant
- designed to withstand operational vibrations and shockproof should you drop the device.
- the **devices can be used in remote locations**, such as **mining and oil sectors, or even in the travel industry, fitted to trucks, trains, and boats, providing a mechanism of easily collecting data and then transferring it back to AWS.**

3. **Storage Aggregation:** The Snowcone and snowball can be used as a way **to aggregate data from multiple sources before shipping it back to AWS for transfer into Amazon S3.**

Example: From within the media and entertainment industry, **get video and audio data from multiple feeds,** especially if you are working in the film industry, this data can then be aggregated to your snowball device and shipped back to AWS for further processing and editing from your wider production team.

4. **Encryption**: This encryption is **backed by keys generated by the Key Management Service (KMS)**. To enhance the security of the device, **the encryption keys are not stored on the device during transit.**

5. enclosure is **Anti-tamper** to specific **verification checks on the boot environment when the device is first switched on**. These measurements and checks help to validate the integrity of the data to ensure that it has not been interfered with at all during transit.

: **The AWS Snowmobile**,

- is used to transfer MASSIVE amounts of data from a single location, up to **100PB per snowmobile** which arrives on a truck as a ruggedized shipping container.

: is only ever operated by AWS personnel
: can also be escorted by an additional security vehicle during the transit of the container to and from premises,
: having GPS tracking available.
: the container is also protected via 24/7 video surveillance systems and alarms.

- looking at migrating **entire data centers to a new location**, or **migrating entire storage libraries or repositories,** and
- it done quickly, securely and cost-efficiently.
- You can **run multiple snowmobiles in parallel** which will allow you to transfer **Exabytes of data**
- AWS snowmobiles if you needed to transfer **more than 10petabytes of data**, anything less than this then you might want to consider using multiple snowball devices.
- Each snowmobile is sent with a connector rack allowing you to **connect it to the backbone network of your own data center**, as a result, this rack comes with up to **2 kilometers of network cabling.**
- designed to operate within ambient temperature **up to 85F (29.4C).**

: If the temperature exceeds this then **an additional auxiliary chiller unit** can be supplied by AWS following a site assessment survey.

: If there isn't sufficient power to feed the snowmobile at the data center then AWS can **also send a separate generator** to power the snowmobile, however, this requires the same space required to home a snowmobile.

- the snowmobile also **encrypts data backed by the Key Management Service**

- Each of the snow devices **uses an E Ink shipping label which is pre-loaded with the delivery details entered on the associated job**. When the snow device leaves AWS premises it can be **tracked via SNS, text messaging and the AWS Management Console.** When the device is prepared to be returned to AWS premises, the **E Ink automatically updates with the appropriate location**
- AWS carries out a secure erase which meets the National Institute of Standards and Technology, more commonly known as **NIST for the sanitization of the media and storage.**

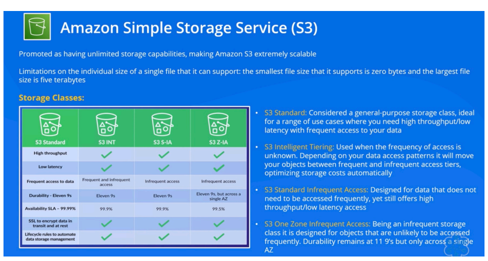
>>>**>**

Amazon AWS Storage summary Points :

>>

1. **Amazon S3**

-> is used for object storage



-> Is highly available,
-> highly durable,
-> very cost-effective,
-> widely accessible.
-> do offer that instant data retrieval
-> more expensive
-> **use cases:** data lakes, data backups, building websites, and much more.
-> Key Elements:

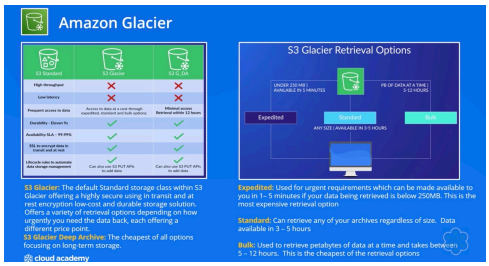
: **storage classes** : **S3 Standard, S3 Intelligent Tiering, S3 Standard Infrequent Access, S3 One Zone Infrequent Access**

2. **Glacier Storage classes**

-> are designed for long-term data storage
-> providing the most cost-optimized solution.

-> Key Elements:

: storage classes : S3 Glacier, S3 Glacier Deep Archive, Expedited, Standard, Bulk



-> For example,

1. if you had a **workload that provided unpredictable pattern access** and looking to **provide a cost-effective storage solution on S3** - use **S3 Intelligent Tiering over Standard**.
2. if you wanted **instant access to objects** for the **lowest cost point** where your data could be **easily reproduced, if lost,** - use **S3 One Zone Infrequent Acces**

: S3 Versioning,

: Lifecycle Rules,
: Transfer Acceleration,
: options to control access to your S3 buckets
- identity-based policies through IAM
- resource-based policies using bucket policies
- S3 Access Control Lists
- in-built public protection settings on the bucket
- Cross Origin Resource Sharing

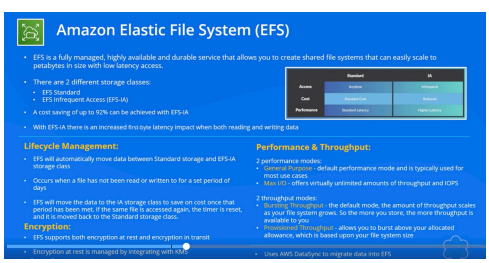
: Basic Security Controls.

-> For Example,

1. given a scenario about how you need to keep **data highly accessible for 90 days** after which **it won't be needed to be accessed anymore, but it will be needed to keep for legal reasons.** - Use **Lifecycle Rules** because this is used to allow to cover from previous versions if changes to your objects are made or if they are deleted. This provides an **automatic method of moving your data between storage classes based on time periods**. So, you can move your data from S3 Standard to S3 Glacier.

3. **The Elastic File System**

-> this is a scalable **network file storage service** for use with Amazon EC2 instances that can easily scale to petabytes
-> **in size with low latency access** providing support to thousands of EC2 instances at once.

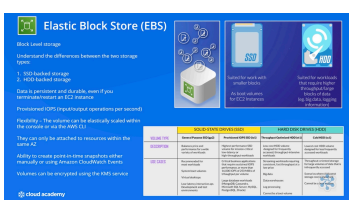


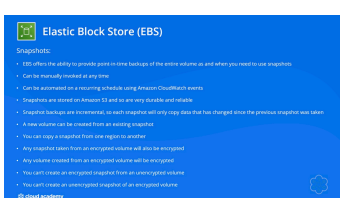
-> EFS is used as **file system storage**.
-> Key Elements:
: storage classes: EFS Standard, EFS Infrequent storage(EFS-IA)
: **performance options**: general purpose, MAX I/O
: throughput modes: Bursting Throughput, Provisioned Throughput
: underlying **architecture from a connectivity perspective:** mount points
: **encryption** with EFS: **at rest** backed by KMS like most AWS services, **in transit** - TLS configured during the mounting process

-> For Example,

1. select the most appropriate performance and throughput mode based on a particular workload.

4. The Elastic Block Store



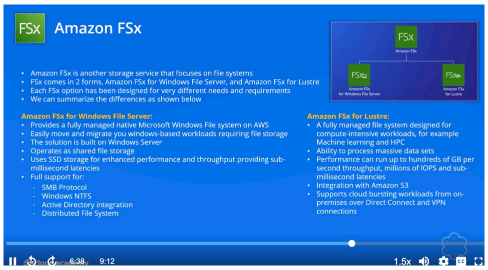


-> It is **persistent** data: the data will not be lost if you terminate the instance that the EBS filling is attached to.
-> It's a really **flexible storage option** for your EC2 instances.
-> Key Elements:
: **EBS snapshots**
- how they work, where they are stored, and also how they work when encryption is applied.
- you can't create an unencrypted snapshot from an encrypted volume.

-> For Example,

1. you have an **unencrypted EBS volume** that now needs to be **encrypted within a different region.** - Cannot do : **take a snapshot of the volume, copy this snapshot to the right region, and then create a new volume from this copied snapshot and select encryption during the volume creation.** Because: you can't create an unencrypted volume from an encrypted snapshot

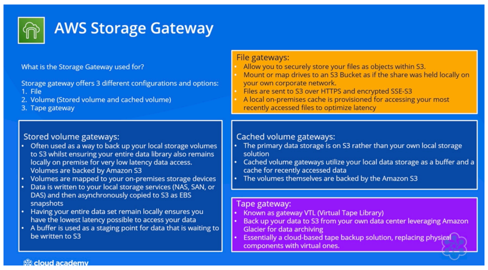
5. Amazon FSx

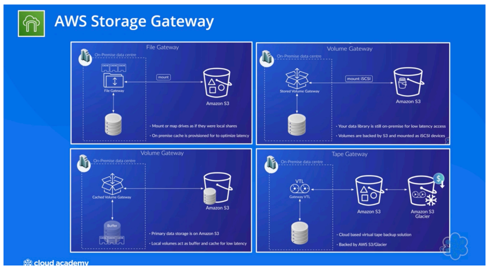


-> it's another file system storage service much like EFS
-> **FSx comes in two flavors:**

- **FSx for Windows** which provides a **Windows File Server** used as a fully managed native Windows File system on AWS, and it uses the **Server Message Block protocol, SMB**.
- **FSx for Lustre** which is a fully managed **Linux-based File System**, and this is designed for **compute intensive workloads and high performance computing.**

6. AWS Storage Gateway.

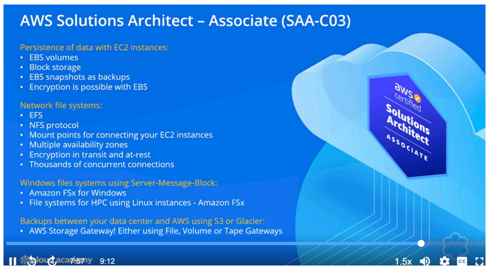




-> Key Elements:
: **use case**.
- a gateway between **your own data center storage systems and Amazon S3, and Glacier**.
- a **hybrid storage** solution with unlimited space.

-> there were three types of gateway: **The file, volume, and tape gateway configuration**.

For Example,



1. About **unlimited object storage** - **S3**
2. if used for **long-term data storage** - **Glacier**.
3. To help with **data management** - How you can use **Lifecycle policies and Versioning**.
4. For **getting data into S3 faster** - **Transfer acceleration.**
5. About **persistence of data with EC2 instances** - **EBS volumes, block storage, EBS snapshots as backups, and encryption is also possible.**
6. Related to **network file systems** - **EFS running the NFS protocol**.
7. **connecting your EC2 instances** - **Mount points**, **multiple availability zones, encryption in transit and at-rest, and thousands of concurrent connections.**
8. Relate to **Windows file systems using the Server Message Block protocol** - **Amazon FSx for Windows,**
9. Relates to **file systems for high-performance computing using Linux instances** - **Amazon FSx Lustre**.
10. talking about **backing up data between your own corporate data center and AWS using S3 Glacier** - **AWS Storage Gateway either using File, Volume or Tape Gateways**.

>>

Lab - Created an Amazon S3 Bucket

- Creating a Bucket Policy in Amazon S3 with IP Address Conditions

- Create a Bucket Policy in S3 with Encryption Conditions

>>>

NOTE: Starting in April 2023, to enable all Block Public Access settings when creating buckets by using the S3 console, you will no longer need the s3:PutBucketPublicAccessBlock permission.

NOTE: Within S3 there are a number of ways to set permissions on your S3 resources.

Bucket Policies

- Bucket policies are applied directly to a bucket within S3 itself
- The bucket policy will restrict anyone from performing any actions within a specific bucket unless their IP address

matches that within the bucket policy's condition statement.

- Bucket policies use the JSON-based (JavaScript Object Notation) policy language.
- A **Policy** is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.
- The **Principal** dictates the user, account or service that the policy will apply to. An asterisk is a wildcard to match all
- **Action:** For S3, there are about 40 individual actions. (CreateBucket, DeleteBucket, etc.) This is where you can select specific actions only.
- **ARNs (Amazon Resource Names)** adhere to the following pattern:
  arn:Paritition:Service:Region:Account-ID:Resource

Partition – This relates to the partition that the resource is found in. For standard AWS regions, this section would be 'aws.'
Service – This reflects the specific AWS service, for example 's3.'
Region – This is the region where the resource is located. Some services do not need a region specified, so this can sometimes be left blank.
Account-ID – This is your AWS Account ID (without hyphens). Again, some services do not need this information, and so it can be left blank.
Resource – The value of this field depends on the AWS service you are using.
For example, if using the Action: "Action":"s3:*", then use the bucket name that you want the permission to apply to arn:aws:s3:::calabs-bucket1np/*

- **Conditions** allow you to define a greater granularity to your policy to only execute under certain conditions and keys.

Example of a policy: The policy denied all S3 actions from any source where the IP address was not 1.2.3.4.

```
{
  "Id": "Policy1681484331853",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1681484329750",
      "Action": [
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::calabs-bucket1np/*",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": "1.2.3.4"
        }
      },
      "Principal": "*"
    }
  ]
}
```

User Policies:

- user policies are set within IAM (Identity & Access Management)

Create a bucket policy which ensures that any uploaded object that does not have server-side AES256 encryption is denied.

- Server-side encryption is about data encryption at rest i.e Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it.
- As long as you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted objects.

Example of a policy : Conditions allow you to define a greater granularity to your policy to only execute under certain conditions and keys.

```
{
  "Id": "Policy1681485181332",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1681485166097",
      "Action": [
        "s3:PutObject"
      ],
```

```
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::calabs-bucket1np/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      },
      "Principal": "*"
    }
  ]
}
```

>>>
Lab - Handling S3 Objects Events With Lifecycle Policies and Server Access Logging
- Setting up lifecycle policies for your S3 buckets
- Creating a server access logging S3 solution
>>>

NOTE: if you are working on a storage solution that needs to better handle the objects, you should consider handling them automatically. For that reason, AWS allowed users to implement the Lifecycle Policies on the S3 buckets.
NOTE:S3 lifecycle policies could be very helpful if you are looking for a solution to automatically perform different kinds of actions on the objects you store inside your bucket.
NOTE: a lifecycle policy allows you to specify actions to perform on objects based on their lifecycle.
NOTE: To have a fully monitored storage solution, you can decide to implement server access logging.
- each operation performed on a single bucket or object will be logged into another bucket.
- not choose the origin bucket as the target bucket for logging as that would create an infinite recursive iteration and it could cost you a lot of money.

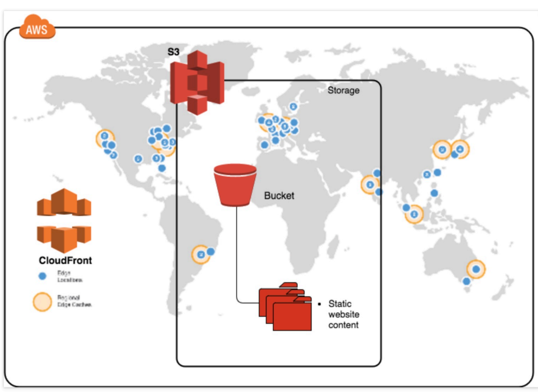
NOTE: Bucket names must be globally unique, regardless of the AWS region in which you create the bucket.
NOTE: Buckets must also be DNS-compliant.
- The rules for DNS-compliant bucket names are:
    1. Bucket names must be at least 3 and no more than 63 characters long.
    2. Bucket names can contain lowercase letters, numbers, periods, and/or hyphens.
    3. Each label must start and end with a lowercase letter or a number.
    4. Bucket names must not be formatted as an IP address (for example, 192.168.1.1).

NOTE: We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Example: it will let objects expire after 90 days and will move objects older than 30 days from the Standard storage class to the One Zone-IA.

NOTE: If you want to have a full understanding of what is happening inside your S3 bucket, you should consider enabling the Server Access Logging functionality.
NOTE: When enabling the server access logs property on a bucket, the target bucket's bucket policy is automatically updated to let the S3 log delivery group (corresponding to the logging.s3.amazonaws.com service principal) put the logs objects

>>>
Lab - Configuring a Static Website With S3 And CloudFront
- Configure static website hosting on Amazon S3
- Configure static websites to work with CloudFront distributions
>>>



NOTE: **static website :**
- You can easily and inexpensively use Amazon Web Services (AWS) to host a website that uses client-side technologies (such as HTML, CSS, and JavaScript) and does not require server-side technologies (such as PHP and ASP.NET).
- is used to display content that does not change frequently.

NOTE: host your static website using the Amazon Simple Storage Service (S3) so that it is
- secure,
- fast,
- protected against data loss, and
- can scale to support enterprise-level traffic.

NOTE: you'll store your website files on Amazon S3 and also use S3

to deliver your content to visitors to your website.
NOTE: use Amazon CloudFront to create a **content delivery network (CDN)**
- A CDN makes your website content available from data centers around the world, called edge locations.
- Using edge locations improves the speed of your website by reducing latency.
- Example : if your website displays large media files such as high-resolution images, audio, or video.

NOTE: The **Properties** tab allows you to enable and disable various Amazon S3 bucket features, including:
- **Bucket Versioning**: Old versions can be kept when objects are updated
- **Default encryption**: A bucket can be configured to encrypt all objects by default
- **Server access logging:** Web-server style access logs can be enabled
- **Requester pays:** When enabled, the entity downloading data from this bucket will pay data transfer costs incurred
- **Static website hosting:** Use this bucket to host a website or redirect requests.

Bucket website endpoint - When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket.
Example: http://calabs-bucket-007.s3-website-us-west-2.amazonaws.com

Example of a policy : This policy will allow public access to all objects in your S3 bucket.
NOTE: This is a permissive policy. In a non-lab environment, security concerns may require you to implement a more restrictive policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AddPerm",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::calabs-bucket-007/*"
    }
  ]
}
```

NOTE: you will download a basic website from a public GitHub repository and load it into your S3 bucket.
NOTE: To be able to access the website, the index.html file must be at the top-level of your Amazon S3 bucket.

Creating an Amazon CloudFront Distribution for the Static Website:

NOTE: Amazon CloudFront is a global Content Delivery Network (CDN) that delivers data securely and efficiently.
NOTE: Origin access control secures S3 origins by allowing access to only designated distributions. This follows AWS best practice of using IAM service principals to authenticate with S3 origins.
NOTE: Amazon CloudFront minimizes end-user latency by delivering content from its entire global network of edge locations.
NOTE: Price Classes let you reduce your delivery prices by excluding Amazon CloudFront's more expensive edge locations from your Amazon CloudFront distribution.
- In these cases, Amazon CloudFront will deliver your content from edge locations within the locations in the price class you selected and charge you the data transfer and request pricing from the actual location where the content was delivered.

NOTE: The time required for deploying a new CloudFront distribution also depends on the number of selected Edge Locations. Selecting all edge locations will take longer to deploy.
NOTE: Amazon CloudFront doesn't always transparently relay requests to the origin. If you did not set a default root object on the distribution you would see an AccessDenied error when you access the CloudFront distribution's domain
NOTE: CloudFront automatically assigns an ID (top of the page) and a Distribution domain name to the distribution and starts updating the edge locations to serve your content:
Example: ID:E26TR42431HK1A :: Distribution domain name : https://d1bljsco6lsyjr.cloudfront.net
NOTE: This domain name is randomly generated and unique for each distribution in Amazon CloudFront.

Policy to S3 has to be updated:

```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
      {
        "Sid": "AllowCloudFrontServicePrincipal",
        "Effect": "Allow",
        "Principal": {
          "Service": "cloudfront.amazonaws.com"
        },
        "Action": "s3:GetObject",
```

```
            "Resource": "arn:aws:s3:::calabs-bucket-007/*",
            "Condition": {
              "StringEquals": {
                "AWS:SourceArn": "arn:aws:cloudfront::063496210328:distribution/E26TR42431HK1A"
              }
            }
          }
        ]
      }
```
NOTE: There are two main types of origin that Amazon CloudFront supports

: **Amazon S3 buckets**

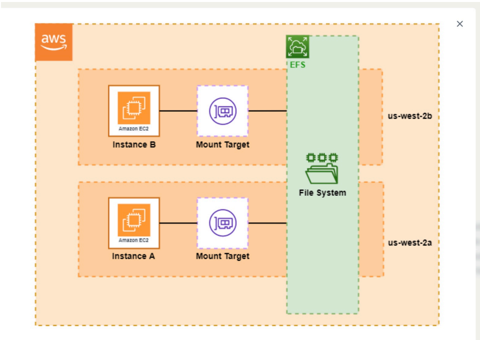
: **custom origin** - A custom origin could be a website being served by an EC2 instance, or it could be a web server outside of AWS.

>>

Lab - Introduction to the Elastic File System

- **Create file systems**
- **Mount file systems to EC2 instances**
- **Read/write files to a file system**

>>



NOTE: Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2 instances in the AWS Cloud.
NOTE: With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files - so your applications have the storage they need, when they need it.
NOTE: Amazon EFS has a simple web services interface that allows you to create and configure file systems quickly and easily.
NOTE:The service manages all the file storage infrastructure for you, avoiding the complexity of deploying, patching, and maintaining complex file system deployments.
NOTE: There are already two instances running in your account. Each instance is in a different Availability Zone, but use the same security group.
NOTE: The way AWS chooses to restrict access to a particular file system is by using security groups

Example:

- Created a security group and added an inbound rule to allow instances to connect to the file systems over port 2049.

EC2-instances-SG : sg-0d9eb95598cc1da2f
NOTE: Security groups associated with NFS file systems need to allow access to port 2049
Filesystem-SG : sg-066887d83b037276f

- Created a file system using AWS Elastic File System service. and mount target to the said above Security group

lab-file-system (fs-00393641e71b923db)
DNS Name : fs-00393641e71b923db.efs.us-west-2.amazonaws.com
Total size : 6.00 KiB
Size in Standard / One Zone : 6.00 KiB (100%)
Size in Standard-IA / One Zone-IA : 0 Bytes (0%)
Lifecycle management
    Transition into IA: 30 day(s) since last access
    Transition out of IA: None

- You have successfully created a new file system, and have created an inbound rule to allow access to the file systems from the two EC2 instances. Now it is time to mount it from an EC2 instance.

: First for the **cloudacademylabs instance** (Instance: i-06207f455b7e21cc6 (cloudacademylabs)) and then repeat once more for **retrieval_instance.** (Instance: i-07ff9f27c1ab90615 (retrieval-instance))

  - Install the NFS client on your EC2 instance using the following command:
    - sudo yum install -y nfs-utils
  - Create a new directory called efs to mount the file system using the following command:
    - mkdir efs
  - Using the NFS client:

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-00393641e71b923db.efs.us-west-2.amazonaws.com:/ efs

- Verify mount

Result:
```
[ec2-user@ip-172-31-25-144 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        468M     0  468M   0% /dev
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           479M  428K  479M   1% /run
tmpfs           479M     0  479M   0% /sys/fs/cgroup
/dev/nvme0n1p1  8.0G  1.8G  6.3G  22% /
tmpfs            96M     0   96M   0% /run/user/1000
[ec2-user@ip-172-31-25-144 ~]$ sudo mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-00393641e71b923db.efs.us-
west-2.amazonaws.com:/ efs
[ec2-user@ip-172-31-25-144 ~]$ df -h
Filesystem                                         Size  Used Avail Use% Mounted on
devtmpfs                                           468M     0  468M   0% /dev
tmpfs                                              479M     0  479M   0% /dev/shm
tmpfs                                              479M  432K  479M   1% /run
tmpfs                                              479M     0  479M   0% /sys/fs/cgroup
/dev/nvme0n1p1                                     8.0G  1.8G  6.3G  22% /
tmpfs                                               96M     0   96M   0% /run/user/1000
fs-00393641e71b923db.efs.us-west-2.amazonaws.com:/  8.0E     0  8.0E   0% /home/ec2-user/efs
[ec2-user@ip-172-31-25-144 ~]$
```

- Update the ownership of the efs folder to the ec2-user using the following command:

```
[ec2-user@ip-172-31-25-144 ~]$ sudo chown ec2-user efs/
[ec2-user@ip-172-31-25-144 ~]$ ls -ltr
total 4
drwxr-xr-x 2 ec2-user root 6144 Apr 14 20:38 efs
```

- Verify Download:

```
[ec2-user@ip-172-31-25-144 ~]$ cd efs
[ec2-user@ip-172-31-25-144 efs]$ wget https://wordpress.org/latest.tar.gz
--2023-04-14 20:50:56--  https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23018887 (22M) [application/octet-stream]
Saving to: ‘latest.tar.gz’

100%[==============================>] 23,018,887  19.3MB/s   in 1.1s

2023-04-14 20:50:57 (19.3 MB/s) - ‘latest.tar.gz’ saved [23018887/23018887]

[ec2-user@ip-172-31-25-144 efs]$ ls -l
total 22480
-rw-rw-r-- 1 ec2-user ec2-user 23018887 Mar 29 17:49 latest.tar.gz


[ec2-user@ip-172-31-6-132 ~]$ cd efs
[ec2-user@ip-172-31-6-132 efs]$ ls -l
total 22480
-rw-rw-r-- 1 ec2-user ec2-user 23018887 Mar 29 17:49 latest.tar.gz
will be already there and no need to download again,
[ec2-user@ip-172-31-25-144 efs]$ du -h
22M     .
```

NOTE: same file on the EFS that was copied earlier from the cloudacademylabs instance. If you can see the file, you have successfully mounted the disk and now you can read and write files to the file system using both EC2 instances. All data in the file system will be persistent and available to all EC2 instances connected to the EFS.

>>

Exam 1+2 on below: Knowledge Check: Storage (SAA-C03)

>>

1. allows and denials in S3 bucket policies :
    A. By default, access is denied to an object, even without an explicit Deny within any policy.
    B. If there is both a Deny and an Allow associated with the principal to a specific object, then access will be denied.
    C. If there is no Deny associated with the principal to a specific object, but there is an Allow, then access will be authorized.
    D. To gain access, there has to be an Allow within a policy that the principal is associated to or defined by within a bucket policy or ACL.
    E. a Deny always takes precedence over an Allow,
2. S3 Intelligent - Tiering provides automatic cost savings for data with unknown or changing access patterns.
3. In order to enable object lock on an S3 bucket, versioning must first be enabled.
4. When configuring an Amazon S3 bucket to monitor for specific events, you can select one of multiple services to send event information. services can you send recorded event information include :
    A. Amazon Simple Notification Service (SNS)
    B. Amazon Simple Queue Service (SQS)
    C. AWS Lambda
5. Any events which are recorded can then be sent to either an SNS Topic, an SQS Queue or a Lambda Function.
6. Selecting the Events tile from bucket properties screen enables you to configure which events are to be monitored.
7. Configuring an S3 bucket for static website hosting requires creating a bucket with the same name as the desired website hostname.
    A. To host a static website you need to configure a bucket for website hosting and then upload the content of the static website to the bucket.
    B. You also need to add an index document. The index document will be the default or home page of your static website. The index documents must be located within your bucket.
8. **Unversioned buckets offer the most cost savings** because one of the costing metrics of S3 is how much data storage you use and using an unversioned bucket means there are not multiple versions of the same file being stored in the bucket.
9. **Versioning enabled is the most costly** because it automatically versions objects unless a command is executed to permanently delete the object version(s) from the bucket.
10. Versioning suspended offers some cost savings because it allows you to suspend further versioning of objects in a bucket; however, it does not initially limit versioning of objects.
11. he types of versioning do not represent equal costs because a versioned bucket and a suspended versioning bucket still contain objects with versions that represent a cost of storage.
12. When a bucket owner PUTs an object in a versioning-enabled bucket, the noncurrent version is not overwritten. Instead, Amazon S3 generates a new version ID and adds the newer version to the bucket.
13. **S3 One Zone - Infrequent Access is for re-creatable,** infrequently accessed data that needs millisecond access.
14. **If you enable versioning on an existing bucket with objects already in them, then their Version ID will be displayed as null** until they have been modified or deleted, at which point they will receive a new Version ID.
15. **Delete is an action you can take on an object in storage as well as a label that appears in brackets once an object is deleted,** but **delete is not a label an unmodified stored object.**
16. "Versioned" and "enabled" refer to bucket versioning labels and properties.
17. When versioning is suspended on a bucket further versioning of objects is prevented.
18. When versioning is suspended you can still see previous object versions in the console, previous object IDs are not lost when versioning is suspended because suspending versioning only affects subsequent objects.
19. Objects created when versioning is suspended on a bucket do not receive a temporary version ID.
20. ACL :
    A. You can of course **use both IAM policies and bucket policies to control access**.
    B. ACLs **do not follow the same JSON format as policies defined by IAM and bucket policies**. Instead, they are far less granular, and different permissions can be applied depending on whether you are applying an ACL at the bucket level or the object level.
    C. Due to the basic structure of an ACL, it is **not possible to implicitly deny access using ACLs.**
    D. you are **not able to implement conditional elements**, like we saw earlier when I mentioned identity-based access.
    E. S3 ACLs allow identities to access specific objects within buckets a different layered approach than bucket policies which are applied at the bucket level only.
    F. ACLs allow you to set certain permissions on each object within a specific Bucket.
21. You have noticed that your S3 bucket contains no CloudTrail logs. Which of the following represent reasons why there are no log files in your bucket?
    A. The log files may be missing from the bucket because object logging has not been enabled by using CloudTrail console, or
    B. there may not have been any API requests made to the objects in the bucket.
22. Timestamps are automatically provided when object level logging is enabled through the CloudTrail console, and any API requests for objects are recorded as new requests and would automatically appear in a bucket once logging is enabled.
23. You can configure notifications to be filtered by the prefix and suffix of the key name of objects.
    A. For example, you can set up a configuration so that you are sent a notification only when image files with a ".jpg" file

name extension are added to a bucket. Or,
   B. you can have a configuration that delivers a notification to an Amazon SNS topic when an object with the prefix "images/" is added to the bucket, while having notifications for objects with a "logs/" prefix in the same bucket delivered to an Amazon Lambda function.
24. When Requester Pays is enabled on an S3 bucket, any costs associated with requests and data transfer become the responsibility of the requester.
25. Anonymous access requests will be denied unless all access is authenticated to your bucket.
26. The bucket owner is still charged for costs associated with the objects stored in the bucket.
27. **Access points only allow you to perform object operations**, for example, S3 GetObject and S3 PutObject. But **it's not possible to use bucket operations, such as S3 DeleteBucket.**
28. With Amazon S3,
   A. you can store as much data as you want and access it when needed.
   B. S3 supports and unlimited number of files in a bucket so it is not necessary to know your storage needs up front or try to estimate.
   C. S3 can be scaled quickly and appropriately to meet the storage demands of your environment.
   D. S3 asynchronously replicates information to all availability zones within a region.
   E. Amazon S3 scales to support very high request rates. If your request rate grows steadily, Amazon S3 automatically partitions your buckets as needed to support higher request rates.
29. If the source bucket has the S3 object lock feature enabled, then the destination must also have it enabled, too.
30. Destination buckets cannot be configured as requester pays buckets.
31. If you want to replicate objects that are encrypted, then you can do so.
32. Once a bucket has been configured for S3 replication, only the objects added to the bucket from that point will be replicated.
33. When configuring **server-access logging**, **the source and target buckets must be in the same AWS Region.** Availability Zones are found within AWS Regions.
34. Log Delivery groups are pre-defined Amazon S3 groups used to deliver log files to your target buckets,
35. Access Control List refers to the Log Delivery group's access to the ACL (Access Control List) of the target bucket.
36. At a high level, **Cross-Origin Resource Sharing (CORS)** allows specific resources on a webpage to be requested from a different domain than its own. And this allows you to build client-side web applications.
37. in Amazon S3 allows specific resources on a webpage to be requested from a different domain than its own, Cross-Origin Resource Sharing allows you to build client-side web applications.
38. **Even though EC2 instance store volumes are part of the EC2 service itself, they are not available for all instance types.**
39. **data deduplication can run as a background process**, which will **not significantly affect the performance** of the file system.
   A. It also is a transparent part of the file system and
   B. will not be obvious to your connected users or clients.
   C. **Data deduplication is automatic** and
   D. will continue to scan your file systems in the background, looking for any extra copies of data.
40. Amazon Elastic File System (EFS) :
   A. It uses standard operating system APIs, so any application that is designed to work with standard operating system APIs will work with EFS.
   B. It supports both NFS versions 4.1 and 4.0, and
   C. uses standard file system semantics such as strong consistency and file locking.
   D. It's replicated across availability zones in a single region, making EFS a highly reliable storage service.
   E. The EFS file system is also regional, and so any application deployments that span across multiple availability zones can all access the same file systems, providing a level of high availability of your application storage layer.
41. It is possible to deploy a single mount target for the entire VPC, but this is not recommended due to added costs for EFS mount targets communicating with instances in different availability zones. The recommended approach **is to deploy a mount target into each availability zone.**
42. The subnet types will not be an issue, and one mount target per EC2 instance defeats the purpose of the service's capability to communicate with a large number of instances simultaneously.
43. **Gateway Virtual Tape Library (VTL)** allows you to back up your data **to S3 from your own corporate data center** and **leverage Amazon Glacier for data archiving.**
44. **Media Changer** is a **virtual device that manages tapes to and from the tape drive to your VTL.**
45. **File gateways** allow you to securely store your **files as objects within S3.** Using as a type of file share which allows you to mount on map drives to an S3 bucket as if the share was held locally on your own corporate network. However, they **related to localized storage rather than virtualized storage.**
46. **Cached volume gateways** store data on Amazon S3 rather than your **own local storage, and they do not use Amazon Glacier for data archiving.**
47. When performing data transfer from on-premises, then we need to configure an agent, a location, and a task.
48. FSx for Windows has 3 price points: **Capacity, Throughput, and Backup**s.
49. much like EFS, there are no setup fees for the use of this service, however, you do pay for the amount of storage capacity that you consume.
   A. This is priced on the average storage provisioned per month and uses the metric of gigabyte-months and offers

varied pricing between a single or multi-AZ deployment.

50. In addition to the actual storage that you use there is also a cost of for the amount of throughput that you configure for your file systems,
    A. this metric is based upon MBps-months.
    B. cost variations exist between single and multi-AZ deployment.
    C. any data transfer costs when using multi-AZ is included in the pricing you see for the multi-AZ deployment.
51. **Cold HDD (SC1)** EBS volume type is ideal for large workloads that are accessed infrequently
    A. The cold HDD volumes offer the lowest cost compared to all other EBS volumes types.
    B. They are suited for workloads that are large in size and accessed infrequently.
    C. Their key performance attribute is its throughput capabilities in megabytes per second.
    D. It also has the ability to burst up to 80 megabits per second per terabyte, with a maximum burst capacity for each volume set at 250 megabytes per second.
    E. It will deliver the expected throughput 99% of the time over a given year, and due to the nature of these volumes, it's not possible to use these as boot volumes for your EC2 instances.
52. **The Standard–IA** storage class reduces storage costs for files that are not accessed every day.
    A. It does this without sacrificing the high availability, high durability, elasticity, and POSIX file system access that Amazon EFS provides.
    B. We recommend Standard-IA storage if you need your full dataset to be readily accessible and
    C. want to automatically save on storage costs for files that are less frequently accessed.
    D. Examples include keeping files accessible to satisfy audit requirements, performing historical analysis, or performing backup and recovery.
    E. Standard-IA storage is compatible with all Amazon EFS features, and is available in all AWS Regions where Amazon EFS is available.
53. **Data Lifecycle Manager** enables you to create policies for EBS volume and EBS-backed AMI creation, retention, and deletion.
54. EBS Recycle Bin allows you to restore snapshots that have been deleted accidentally.
55. AWS Backup allows a company to create and enforce policies around backing up EBS volumes, which deals with creating the snapshots, not managing or undoing their deletion.
56. AWS Config creates policies around creating resources, their approved configuration, and what if any changes are made to existing resources.
57. AWS Resource Groups allows you to manage numerous related resources as a single resource.
58. You have decided to use AWS Storage Gateway, but want all data within the gateway to be retrievable to on-premise employees with minimal latency. You've decided to use block storage that backs up EBS snapshots. Which type of storage gateway would best suit you?
    A. Both file gateways and cached volume gateways provide local caches to store frequently accessed data.
    B. Stored volume gateways keep all files locally, so all stored data can be retrieved with low latency, and so is the best option in this case.
59. AWS Backup allows a company to create and enforce policies around backing up numerous data services, and centralizes management of several AWS accounts.
60. An instance store provides temporary block-level storage for your instance.
    A. This storage is located on disks that are physically attached to the host computer.
    B. 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.
    C. Data in the instance store is lost under the following circumstances:
        a. The underlying disk drive fails
        b. The instance stops
        c. The instance terminates
    D. If the instance reboots (either intentionally or unintentionally) the data persists.
    E. An Amazon EC2 Instance Store provides temporary block-level storage for your instance.
    F. An instance store is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content.
    G. Ephemeral storage is ideal for non-persistent data.
61. AWS Backup uses backup features from existing services, so for example, if you were to manage your EBS backups, AWS Backup would manage these through the EBS Snapshot feature as a way of performing the backup.
62. When performing data transfer from on-premises, then we need to configure an agent, a location, and a task.
    A. The agent will be used on the customer side, so it sits outside of AWS, and it's just a virtual machine supported by VMware ESXi, KVM, or Microsoft Hyper-V hypervisors, so it should be compatible with your existing infrastructure.
    B. The location identifies the endpoint of a DataSync task. So as a result, every time you create a DataSync task you will need to specify the source location and the destination location, dictating where you want to move data from and to.
    C. The task contains the details of the operation that you are trying to carry out and perform with DataSync, so it will contain the locations that were created and specified for both the source and destination, in addition to the configuration and conditions of how the data transfer will take place.
    D. Before I move on, I just want to highlight that the DataSync tasks will only copy your storage data; it doesn't include any file systems permissions or settings.

63. Provisioned IOPS SSD volume (IO1)EBS volume type is ideal for applications requiring I/O intensive workloads
    A. Provisioned IOPS SSD volumes deliver enhanced predictable performance for applications requiring I/O intensive workloads.
    B. They also have the ability to specify IOPS rate during the creation of a new EBS volume. And when the volume is attached to an EBS-optimized instance, EBS will deliver the IOPS defined and required within 10%, 99.9% of the time throughout the year.
    C. And the volumes range from four to sixteen terabytes in size. Per volume, the maximum IOPS possible is set to 20,000 IOPS.
64. EFS also provides two different throughput modes, and throughput is measured by the rate of mebibytes.
    A. The two modes offered are Bursting Throughput and Provisioned Throughput.
65. With the Bursting Throughput mode, which is the default mode, the amount of throughput scales as your file system grows.
    A. So the more you store, the more throughput is available to you.
    B. The default throughput available is capable of bursting to 100 mebibytes per second.
    C. However, with the standard storage class, this can burst to 100 mebibytes per second per tebibyte of storage used within the file system.
66. Provisioned Throughput mode.:
    A. If you are finding that you're running out of burst credits too often, you might need to consider using the Provisioned Throughput mode.
    B. Provisioned Throughput allows you to burst above your allocated allowance, which is based on your file system size.
    C. So if your file system was relatively small but the use case for your file system required a high throughput rate, then the default bursting throughput options may not be able to process your request quickly enough.
67. The Backup plan feature designates the frequency, storage, replication, and tagging of backups managed by AWS Backup
68. When using AWS Backup you will need to create backup policies or backup plans.
    A. These simply determine the exact requirements that you need for your backups and contain information such as:
        a. A backup schedule
        b. Backup window
        c. Lifecycle rules, such as the transition of data to cold storage after a set period
        d. A backup vault, which is where your backups are stored and encrypted through the use of KMS encryption keys
        e. Regional copies
        f. Tags
    B. Once you have created your backup plans, you can assign resources to them.
    C. This allows you to create multiple backup plans each with different criteria to meet the backup needs of different types of resources.
    D. Through the use of tags, you can associate multiple resources at once using tag-based backup policies, this ensures you capture all of the required resources at once within your plan.
69. Amazon EFS offers two performance modes: General Purpose mode and Max I/O mode.
    A. The performance mode is selected when the file system is created.
    B. The Max I/O performance mode is best suited for applications where multiple EC2 instances access the file system, as it can scale to higher levels of aggregate throughput and operations per second with a tradeoff of negligibly higher latencies for file operations.
70. AWS DataSync is a service that allows you to easily and securely transfer data from your on-premises data center to AWS storage services.
    A. It can also be used to manage data transfer between two different AWS storage services, too.
71. The Snow family consists of a range of **physical hardware devices** that are all designed to enable you to transfer data into AWS from the edge or beyond the cloud, such as your data center, but they can also be used to transfer data out of AWS too, for example, from Amazon S3 back to your data center.
72. You need a shared file system that integrates with Amazon S3, supports multiple connections simultaneously, and is ideal for compute-intensive workloads, such as high-performance computing. Amazon FSx for Lustre AWS service should you use.
73. **Amazon FSx for Lustre** is a fully managed **file system designed for compute-intensive workloads,**
    A. for example, Machine Learning and high-performance computing.
    B. It has the ability to **process massive data sets.**
    C. Performance can run up to hundreds of GB per second of throughput, millions of IOPS, and sub-millisecond latencies.
    D. It has integration with Amazon S3 and supports and supports cloud-bursting workloads from on-premises over Direct Connect and VPN connections.
74. **File gateways and tape gateways** are **directly connected to Amazon S3**, and Amazon Glacier by extension.
    A. As such, each gateway has **unlimited total storage capacity**, although other factors limit the workload these gateways are able to support.
75. Volume gateways, as the name implies, are stored on block storage volumes, similar to Amazon EBS.
    A. The volumes offer a limited amount of storage, even though the volumes can be stored in Amazon S3.
76. Virtual tape for the tape gateways has a maximum size of 5 TB.
77. Both file gateways and cached volume gateways provide local caches to store frequently accessed data.
78. Stored volume gateways keep all files locally, so all stored data can be retrieve with reduced latency.
79. Tape gateways are an archival method, and not ideal for data that needs to be readily available.

80. **AWS Backup uses backup features from existing services,**
    A. so for example, **if you were to manage your EBS backups, AWS Backup would manage these through the EBS Snapshot feature as a way of performing the backup.**
    B. **It uses each service's specific backup method.**
81. It is possible to deploy a single mount target for the entire VPC, **but this is not recommended due to added costs for EFS mount targets communicating with instances in different availability zones.** The recommended approach is to **deploy a mount target into each availability zone.** The subnet types will not be an issue, and one mount target per EC2 instance defeats the purpose of the service's capability to communicate with a large number of instances simultaneously.
82. In testing your new EFS file system, your application's **baseline workload quickly exhausts the standard EFS mebibyte per second limits. Change from Bursting Throughput mode to Provisioned Throughpu,** configuration adjustments would allow EFS to provide unlimited mebibytes per second for as long as your application requires
83. As a member of the data management team, you are reviewing which Amazon EFS storage classes for the company's various data types.
    A. Application backup files need to be stored but only accessed in the event of a critical failure, which rarely occurs.
    B. The files are critical to system recovery, and it is important to mitigate data loss as much as possible.
    C. then, **EFS Standard-Infrequent Access (IA) EFS storage class would be most effective for storing these application backup files**
84. **The Standard–IA storage class reduces storage costs for files that are not accessed every day. It does this without sacrificing the high availability, high durability, elasticity, and POSIX file system access that Amazon EFS provides.**
85. **We recommend Standard-IA storage if you need your full dataset to be readily accessible and want to automatically save on storage costs for files that are less frequently accessed.**
    A. Examples include keeping files accessible to satisfy audit requirements, performing historical analysis, or performing backup and recovery.
86. Standard-IA storage is compatible with all Amazon EFS features, and is available in all AWS Regions where Amazon EFS is available.
87. The **EFS file system is also regional**, and **so any application deployments that span across multiple availability zones can all access the same file systems, providing a level of high availability of your application storage layer.**
88. different methods of using policies and permissions to gain access to S3 resources (S3 buckets and objects within those buckets): We can control access to S3 resources via policies using both
    A. identity-based policies
    B. resource-based policies
89. Different security controls and methods that have been built into Amazon S3:
    A. **resource ownership:**
        a. Resources in S3 can be defined as **buckets and objects**.
        b. By default when an Amazon bucket is created or an object is uploaded to Amazon S3 within an account, then that **AWS account becomes the owner of that resource.**
        c. So resource ownership is **managed at the account level.**
        d. Example: with the correct permissions applied, allow another AWS account to upload objects to one of your own buckets, and your account would be the resource owner of that bucket. However, when a different AWS account uploads an object within that same bucket, the AWS account that performs the upload of that resource becomes the resource owner of that object.
        e. **the bucket owner does not become the resource owner of the object,** and in addition to this, **the bucket owner would not have access to these objects either that have been uploaded by another account.**
            1. You can either accept the **default option of the object writer,** maintaining the object's ownership, alternatively **select bucket owner preferred for the owner of the bucket to obtain ownership of any objects uploaded to the bucket.**
            2. you must **update the bucket policy to enforce all Amazon S3 put operations to include the bucket-owner-full-control canned ACL.**
        f. The **bucket-owner-full-control canned ACL applies to objects only.**
        g. The benefits of enforcing the bucket owner to assume control over objects uploaded by another account allows the bucket owner **to maintain a level of access control over all the objects within a bucket.**
        h. This also helps **to simplify the management of the objects residing in the bucket.**
    B. **identity-based policies:**
        a. Identity-based policies are **attached to the IAM identity requiring access,** and then **using IAM permission policies, either in-line or managed, they can then be associated to the user, a group that the user belongs to, or via a role that the user has permission to assume.**
        b. **Identity-based policies define the resource in the policy.**
        c. Example: You can of course use conditions within your policies to manage and refine access to your resources even further. For example, in this policy, the identity would be able to perform the actions CreateBucket, DeleteBucket, and PutObject on the s3deepdive bucket on the condition that their source IP address was within the range of the CIDR block 10.1.0.0/1
        d. There is **on need to add the Principle element**
        e. use IAM if you want to **centrally manage your access control methods all in the one service**, that being IAM.
        f. if you **have multiple permissions to apply to a large number of buckets,** the management of doing so might

be easier to do from within IAM access across one or two policies rather than creating separate bucket policies that you would have to attach to each of your buckets.

g. IAM has the added advantage of being **able to control access for more than one service at a time within its policies**

h. IAM policies can be
   1. a maximum of **2 kilobytes in size for users,**
   2. **5 kilobytes for groups**, and
   3. **10 kilobytes for roles.**

i. **Identity-based policies define the resource in the policy.**

C. **resource-based policies:**

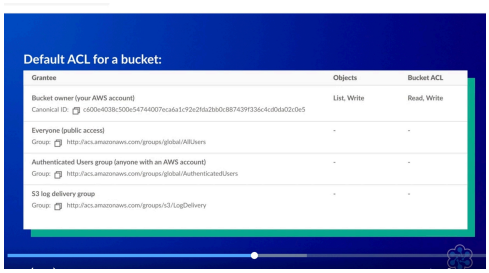
a. the policy is associated with a resource rather than the identity.

b. resource-based policies come in the form of **Access Control Lists** and **bucket policies**. Because of this, you need to define within the policy who will be allowed or denied access.

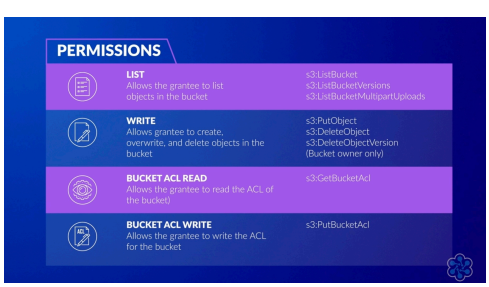
c. This is managed differently depending on if **you're using bucket policies or Access Control Lists.**

d. **Bucket policies :**
   1. A bucket policy is written in **JSON** and is directly attached to your bucket as another means of granting and restricting access control.
   2. **by default, when you create a bucket, no bucket policy exists.**
   3. To add a modified bucket policy, you must select the bucket, go to permissions, and then select the Edit button in the bucket policy section:
      A. You can either freely write your own bucket policy in the JSON window,
      B. view some policy examples, or
      C. use the AWS policy generator to help you create your bucket policy.
   4. When working with bucket policies, you must specify a **principle element,** which defines the principal who is associated with the action and effect defined in the statement.
   5. the permissions defined within the statements of a bucket policy apply to the bucket and the objects that reside within the bucket.
   6. bucket policies **only control access to the S3 bucket and its objects.**
   7. use bucket policies if you want **to maintain your security policies within S3 alone.**
   8. You can **also grant cross-account access using bucket policies without having to create and assume roles that are created within IAM.**
   9. bucket policies can reach a size of **20 kilobytes.**

e. **Aaccess control lists(ACL) :**
   1. allow you to control **access to buckets in addition to specific objects within a bucket by groupings and AWS accounts.**
   2. you can set different permissions per object, you can specify different permissions for different predefined S3 groups defined by the Grantee column.

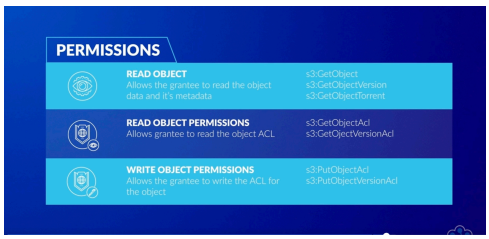


   3. A canned ACL **is a predefined grant that contains both grantees and permissions.**
   4. **Permissions** are based on the grantee of which there are four:
      A. **The bucket owner.** This'll be your own AWS account and will have full control over all objects and the bucket itself.
      B. **Everyone (public access).** Permissions set against this grantee would mean **anyone can access using the permissions applied, providing the object had been made public**. These **requests can be signed, authenticated, or unsigned, unauthenticated.** Everyone group **has read access to the object.**
      C. **Authenticated users.** This option will allow IAM users from any AWS account to access the object via signed request, authenticated.
      D. **S3 log delivery group.** This is a group used to deliver log files when server access logs has been configured, and the bucket is used to store and write log files to

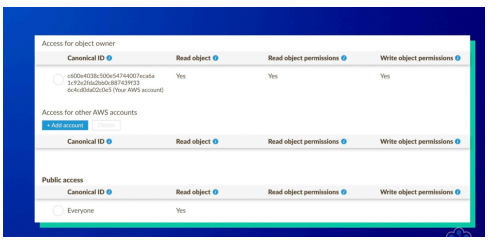


   5. For clarification about what these permissions actually grant, they are as follows:
      A. **List**, which allows the grantee to list objects in the bucket.
      B. **Write**, which allows a grantee to create, overwrite, and delete objects in the bucket.
      C. **Bucket ACL Read,** which allows the grantee to read the ACL of the bucket.
      D. **Bucket ACL Write,** which allows the grantee to write the ACL for the bucket.
   6. **ACLs do not follow the same JSON format as policies defined by IAM and bucket policies.**

7. they are far less granular, and different permissions can be applied depending if you are applying an ACL at the bucket level or the object level.



8. **it is not possible** to **implicitly deny access using ACLs;** Neither are you able to **implement conditional elements,**
9. **ACL looks like at the bucket level:** you can also add access for another account as a grantee. To configure this access, you'll be required to **enter the canonical ID of the AWS account that you would like to have access.**
10. **ACL looks like at the object level:** by default, **the resource owner has full control over the object.** you can add access for another account. Need to add the appropriate canonical AWS account ID and the relevant permissions.



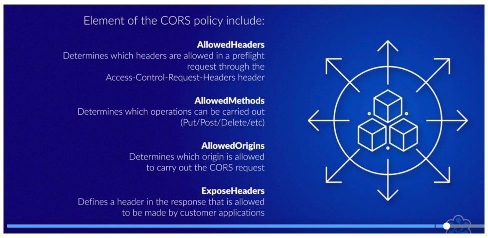
A. There is **no write permission when working with object ACLs**, unlike bucket ACLs.
B. **Read object** allows the grantee to read the object data and its metadata.
C. **Read object** permissions allows the grantee to read the object ACL.
D. **Write object** permissions allows the grantee to write the ACL for the object.

D. How is this access governed if there are conflicting permissions to the object in the bucket that they are trying to access?
   a. All of these policies will be viewed together to determine the resulting access and **will handle any permission conflict in accordance with the principle of least privileged.**
      1. by default, AWS states that access is denied to an object, even without an explicit Deny within any policy.
      2. To gain access, there has to be an Allow within a policy that the principal is associated to or defined by within a bucket policy or ACL.
      3. If there was no Deny defined, but there is an Allow within a policy, then access will be authorized.
      4. However, if there is a single Deny associated with the principal to a specific object, then even if an Allow does exist, this explicit denial will always take precedence overruling the Allow and access will not be authorized.
      5. a Deny always takes precedence over an Allow

E. **Amazon S3 access points**: **how to scale access to shared buckets using S3 access points:**
   a. Managing access to this data through the use of IAM policies, bucket policies, and ACLs can lead to an administrative burden when working with access at scale, where each requester might require different permissions to your data, which can lead to large policies which can be confusing to decipher.
   b. AWS released **Amazon S3 access points**, which helped to simplify the management of controlling and managing shared data at scale on S3.
   c. S3 access point can be created configured and attached **to a single bucket.**
   d. You **can't use the same access points across multiple buckets,** but **you can have multiple access points attached to a single bucket.**
   e. Access Points **only allow you to perform object operations only**, for example, S3 GetObject and S3 PutObject. But **it's not possible to use bucket operations**, such as S3 DeleteBucket.
   f. Permissions assigned to each S3 access point work in conjunction with the underlying bucket policy, allow you to configure different permissions for different access points.
   g. As a part of the access point configuration, you can decide if you **only want to accept requests from a specific VPC**, and therefore restrict access to your own private network. Alternatively, you **can allow access from anywhere outside of your VPC.**
   h. **By default,** for all newly created access points, **all public access is blocked.**
   i. your access point restrictions effect only those connections to your bucket that are accessed via the access point.
   j. **Access Point Policy.** And this allows you to add a JSON policy to define permissions when using the access point.
   k. When granting permissions within the access point policy, they can only be as permissive as the associated underlying bucket policy allows.
   l. Example:This means that if the user Stuart was given permissions of S3 PutObject with an access policy, then the underlying bucket policy would also need to grant your access of S3 PutObjects.
   m. As a result, if you're going to use access points to help you maintain security at scale for your shared data set, then it is best to add a policy to the bucket to delegate access control to the access points.
   n. it's not possible to have two access points with the same name in the same region.
   o. Once you have created an access point for your bucket with predefined security controls, you can connect to your bucket access point via the AWS Management Console, AWS SDKs or the S3 REST APIs.

F. **how to manage public access to your buckets:**
   a. When creating a new bucket in S3, there's an option that's dedicated to helping you protect your bucket from public access. And by default, you can see that there's a checkbox that's ticked, which blocks all public access.
   b. if I tried to allow any kind of public or cross account access for the bucket policy or ACL, then access would still not be allowed as the bucket still has the block all public access setting enabled.
   c. allow some public access based on certain security controls and block others:
      1. you can block public access to buckets and objects granted through new access controllers,
      2. block public access to buckets and objects granted through any access controllers,
      3. block public access to buckets and objects granted through new public bucket or access point policies,
      4. block public and cross account access to buckets and objects, through any public bucket or access point policy.

G. **S3 Alerting with Access Analyzer**
   a. The Access Analyzer is designed to alert you when any of your S3 buckets have been configured to allow either public access or buckets with access from other AWS accounts including third-party AWS accounts.
   b. If you have any buckets that are configured to allow this access then Access Analyzer will identify which buckets they are, what level of access has been granted and how that access is being given.
   c. Access Analyzer **updates findings every 30 minutes** and you can download a report containing all the bucket information within that region and the public access or cross account access that has been configured.
   d. to use Access Analyzer within your regions, you must first create an account level analyzer in IAM for each region that you want to review.
   e. Need to go to IAM to set up an Access Analyzer for this region.

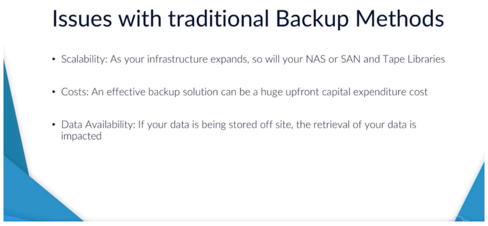


H. **cross-origin resource sharing with S3 [CORS]**
   a. At a high level, **CORS allows specific resources on a webpage to be requested from a different domain than its own.**
   b. this allows you to build client-side web applications.
   c. you can also utilize CORS support to access resources stored in S3.

>>>

Data backups (DR)

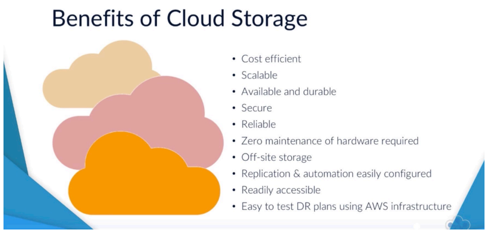
>>



-> If using a tape backup method, then the tapes could be faulty, due to overuse, and the data become unreadable, or even lost in transit.

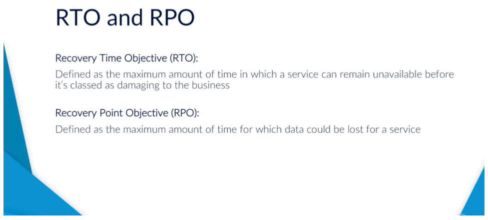
-> There may also be manual activity involved, from tape rotation to tape labeling, which may result in specific data not being backed up or not being able to find the right data.

-> As your infrastructure expands, so will your NAS or SAN and tape libraries. This can be very costly to maintain and implement.



-> You could perform the following steps to quickly recover from the disaster.

: You could launch a new environment with new instances, based on your own custom AMIs, complete with custom applications within a VPC.
: Then you could create EBS storage volumes, based on your backed up data, from within AWS, and attach these volumes to those instances.
: You now have your application servers up and running, with the latest production data.
: Couple this with some minor networking components, and you could then communicate with other sites you may have running.



-> **Recovery Time Objective, RTO**, is defined as the maximum amount of time in which a service can remain unavailable before it's classed as damaging to the business.

-> **Recovery Point Objective, RPO**, is defined as the maximum amount of time for which a data could be lost for a service

: You may just be using AWS as a backup solution, and retaining all of your production data on-site at your own data center.
Or
: you might be operating entirely within AWS, and utilizing AWS storage services for data backup of your AWS environment, such as snapshots of EBS volumes, or of RDS instances.

-> method on which you choose to move your data from on-premise into the cloud can vary depending on your own infrastructure and circumstances:

: **If you have a Direct Connect connection to AWS,** then you can use this to move data in and out of the environment, which can support connectivity of up to 10 gigabits per second.

: If **you don't have a direct connection link between your data center and AWS** then you may have a hardware or software VPN connection which could also be used

: if **you don't have either of these as connectivity options** then you can use your own internet connection from the data center to connect and transfer the data to AWS.

: **Depending on how much data you need to move or copy to AWS**, there are physical disc appliances that are available,

-> using the **AWS Snowball service**, whereby AWS will send you an appliance, either 50 Terrabytes or 80 Terrabytes in size, to your data center, where you can then copy your data to it before it is shipped back to AWS for uploading onto S3.

: You can use multiple **Snowballs** at a time to transfer Petabytes of data if required.

: AWS does offer an even larger storage transfer solution known as **Snowmobile.** This is an Exabyte-scale data transfer service, where you can transfer up to 100 Petabytes per Snowmobile, which is a 45-foot long shipping container pulled by a semi-trailer truck.

-> The **AWS Storage Gateway service** is another method which acts as a gateway between your data center and your AWS environment.

: A software appliance is configured on-site at your data center and offers a range of options in moving data into AWS.

-> **Durability.** When looking at the durability of a data backup, you'll need to ascertain the criticality of that data to ensure it offers the most suitable resiliency and redundancy. For example, if I look at the Amazon S3 service it has the following classes available.

: **The Standard Class,** which provides 11 nines of durability and 4 nines of availability.

: **The Infrequent Access Class, known as IA,** provides 11 nines of durability, but only 3 nines of availability, and this is often used as a backup store over the Standard Class.

: **Amazon Glacier.** This also provides 11 nines of durability, and is used as a cold storage solution. This offers the cheapest storage cost of the three, but does not allow immediate access to the files.

S3 Standard offers

-> eleven nines of durability, and four nines availability over a given year.

-> This reliability is also coupled with an SLA of the service.

-> This level of durability is achieved by the automatic replication of your data to multiple devices and multiple availability zones within a specified region.

-> It also has a number of security features, enabling it to support encryption both in transit, and when data is at rest with both client and server side encryption mechanisms.

-> It also offers data management capabilities through the use of lifecycle policies that can automatically move data to another storage class for cost optimization, or it can delete the data all together depending on your lifecycle policy configuration.

The Standard Infrequent Access class also offers

-> the eleven nines durability, but only three nines availability, whereas the Standard offered four.

-> This class however, is predominantly **used for data that is accessed less frequently than the Standard**

-> As a result, the availability takes a hit, but in doing so, **the cost for this class is far less than the Standard class.**

-> This makes it an effective choice **when using S3 to store backup data for DR purposes**,

:as you're still getting the eleven nines durability, and

:have high speed data retrieval access as and when you need it.

:This class also adheres to the same SLA listed previously,

:the same encryption and data management mechanisms.

-> The main difference here is the cost of the actual storage itself.

Amazon Glacier.

-> it operates in conjunction with S3, and is considered the **third storage class of S3**.

-> Amazon Glacier **stores data in archives** as opposed to S3 buckets, and these archives **can be up to 40TBs in size.**

->The archives are **saved within Vaults,** which act as containers for archives.

-> Specific security measures can be applied to these Vaults to offer additional protection of your data.

-> This class is essentially **used for data archiving and long-term data retention**, and is commonly referred to as the **cold storage service within AWS.**

-> When you need to move data into Glacier,

: You can use the **lifecycle rules within S3 to move data from a Standard or Infrequent Access Class directly to Amazon Glacier.**

: You can use the AWS SDKs, or the underlying Amazon Glacier API.

-> Amazon Glacier also retains the same level of durability as the other two classes, and also offers support for both encryption in transit and at rest to protect sensitive information.

-> it also enforces its **own security measures** that differ from S3, such as

: **Vault Locks,** enabling you to enforce a write once, read many, or

: **WORM control**,

: vault access policies, which provide access to specific users or groups.

-> Amazon Glacier can also be used to help **comply with specific governance controls**, such as, HIPPA and PCI, within an overall solution.

-> If you need to retrieve data from Amazon Glacier, than depending on your retrieval option, it can take anything between a number of minutes to a number of hours for that data to be made available for you to then explore.

: **Expedited.** This is **used when urgent access is required to a subset of an archive,** which is l**ess than 250MB,** and the data will typically be available within **1-5 minutes.** The cost of this is **$0. 03 per GB and $0. 01 per request.**

: **Standard.** This allows you to **retrieve any of your archives, regardless of size**, and it will take between **3-5 hours** to retrieve the data. As you can see, the cost of this method is **cheaper than Expedited.**

: **Bulk.** This is the **cheapest option for data retrieval,** and can retrieve **petabytes of archive dat**a. This option normally takes between **5-12 hours,** and as you can see, it is significantly cheaper than the other options.

-> **S3 offers** a great level of durability and availability of your data, which is perfect for DR use cases. However, if from either a business continuity or compliance perspective, you had a requirement to access S3 in multiple regions, than **the default configuration of having your data stored within a single region**, **and copied across multiple AZ's is simply not enough.** In this scenario, you would **need to ensure you configured Cross Region Replication.**

-> **By default, S3 will not copy your data across multiple regions**, it has to be explicitly configured and enabled.

-> **Multipart** upload essentially allows you to upload a single large object by breaking it down into smaller contiguous chunks of data. The different parts can be uploaded in any order, and if there is an error during the transfer of data for any part, than only that part will need to be resent. Once all parts are uploaded to S3, S3 will then reassemble the data for the object.

: There are a number of benefits to multipart upload. These being, **Speed and Throughput.**

: As mutliple parts can be **uploaded at the same time in parallel,** the speed and throughput of uploading can be enhanced.

: **Interruption Recovery.** Should there be any transmission issues or errors, it will only affect the part being uploaded, unaffecting the other parts. When this happens, only the affected part will need to be resent.

: **Management.** There is an element of management available whereby you can, if required, pause your uploads and then resume them at any point. Multipart uploads do not have an expiry time, allowing you to manage the upload of your data over a period of time.

-> **S3 Security:** Some of these security features, which can help you maintain a level of data protection are:

- IAM Policies. These are identity and access management policies that can be used to both allow and restrict access to S3 buckets and objects at a very granular level depending on identities permissions.

- Bucket Policies. This are JSON policies assigned to individual buckets, whereas IAM Policies are permissions relating to an identity, a user group, or role. These Bucket Policies can also define who or what has access to that bucket's contents.

- Access Control Lists. These allow you to control which user or AWS account can access a Bucket or object, using a range of permissions, such as read, write, or full control, et cetera.

- Lifecycle Policies. Lifecycle Policies allow you to automatically manage and move data between classes, allowing specific data to be relocated based on compliance and governance controls you might have in place.

- MFA Delete. Multi-Factor Authentication Delete ensures that a user has to enter a 6 digit MFA code to delete an object, which prevents accidental deletion due to human error.

- Versioning. Enabling versioning on an S3 bucket, ensures you can recover from misuse of an object or accidental deletion, and revert back to an older version of the same data object. The consideration with versioning is that it will require additional space as a separate object is created for each version, so that's something to bear in mind.

>>>