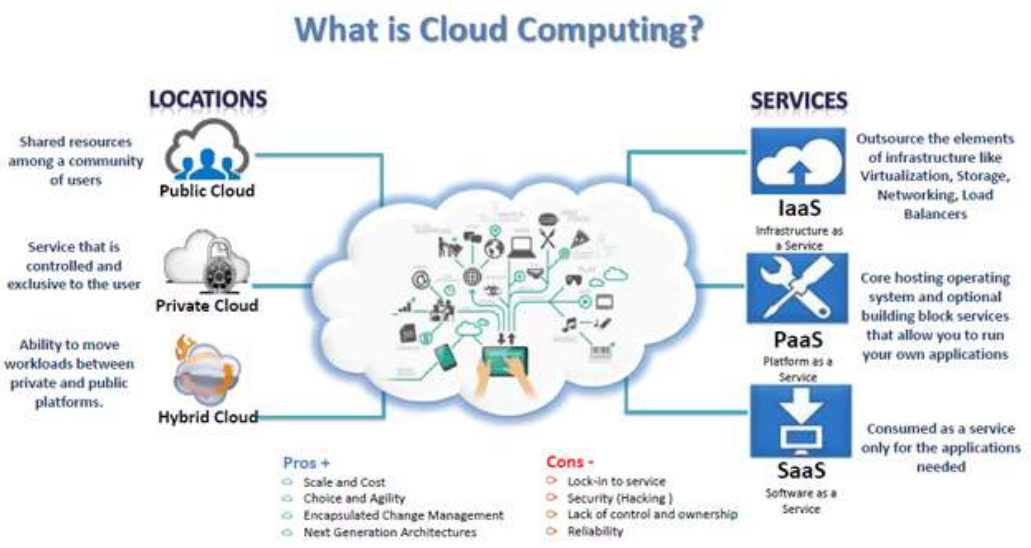
AMAZON WEB SERVICES

AWS Services for Data Analytics

Basic Concepts & Terminology

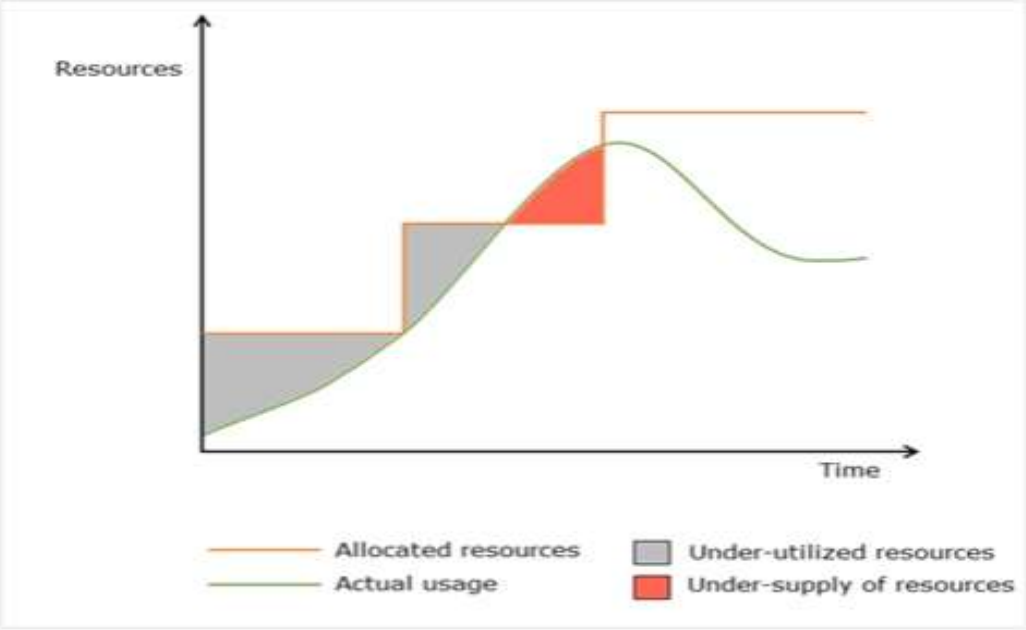
Understanding Cloud Computing

Cloud Computing is a term that indicates delivery of computing resources such as servers, storage, network etc. over the internet by a service provider and is accessible from anywhere and at any time.

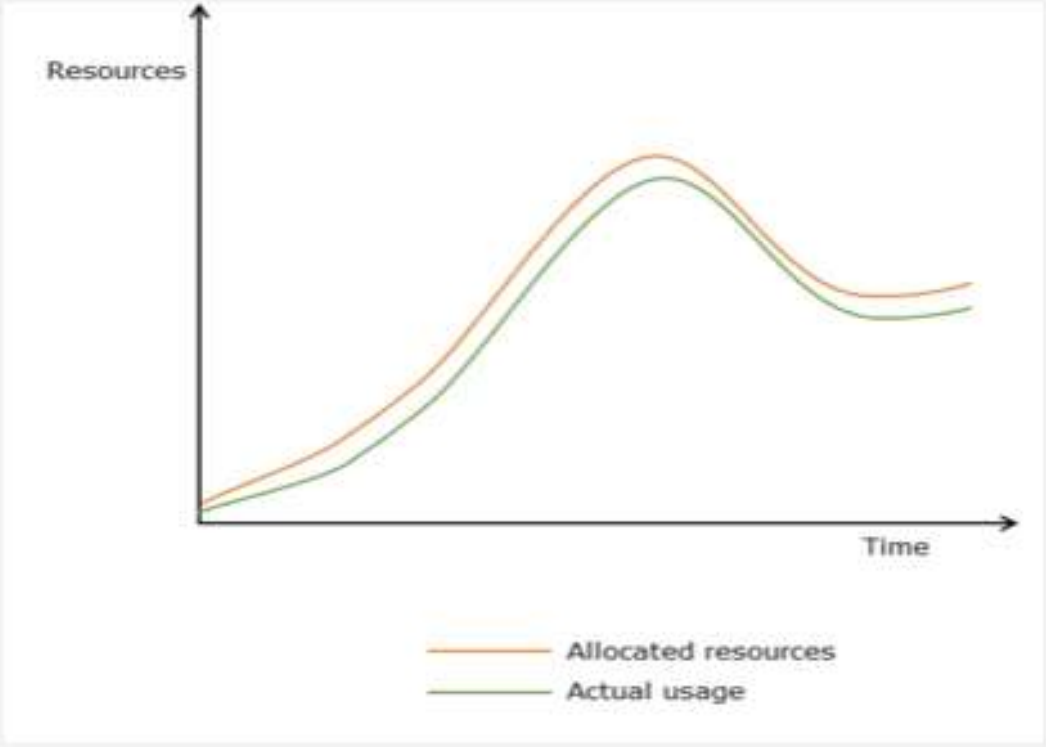


Problems with traditional computing model

- Under-utilization or Over-supply
- Not cost effective

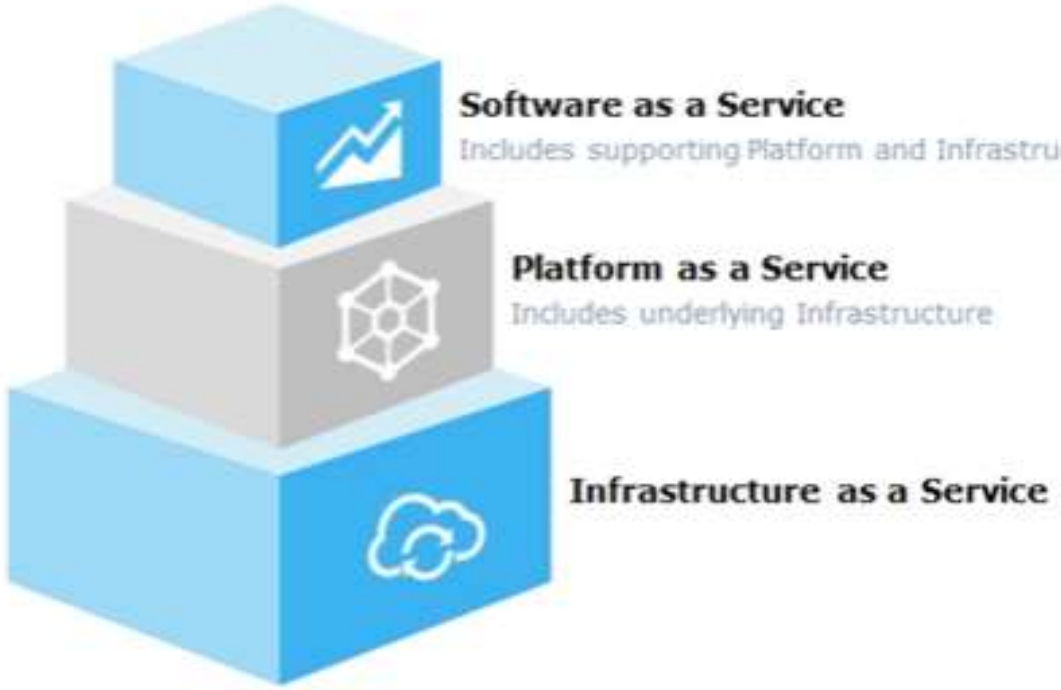


Runtime provisioning with cloud



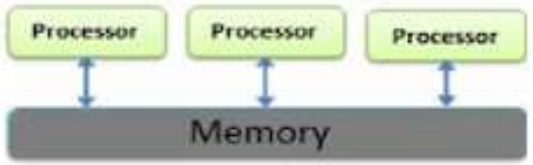
Benefits of cloud computing

Cloud computing service models

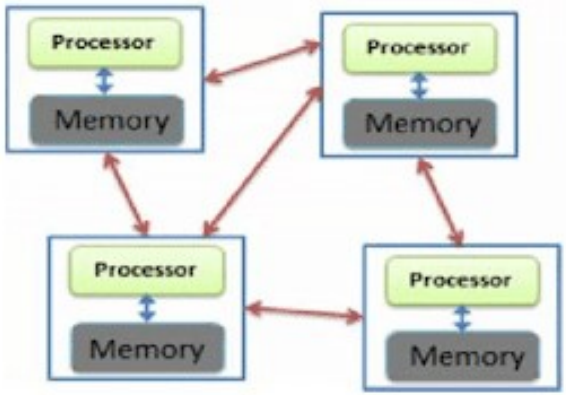


Parallel Computing & Distributed Computing

- Parallel computing speeds up a computational task by dividing it into smaller jobs across multiple processors inside one computer.



- Distributed computing uses a distributed system, such as the internet, to increase the available computing power and enable larger, more complex tasks to be executed across multiple machines.



Parallel Computing vs. Distributed Computing

Number of Computers Required

- Parallel computing typically requires one computer with multiple processors.
- Distributed computing involves several autonomous and often geographically separate computer systems working on divided tasks.

Parallel Computing vs. Distributed Computing

Scalability

- Parallel computing systems are less scalable than distributed computing systems because the memory of a single computer can only handle so many processors at once.
- Distributed computing systems can always scale with additional computers.

Parallel Computing vs. Distributed Computing

Memory

- In parallel computing, all processors share the same memory and the processors communicate with each other with the help of this shared memory.
- In Distributed computing, systems have their own memory and processors.

Parallel Computing vs. Distributed Computing

Synchronization

- In parallel computing, all processors share a single master clock for synchronization.
- In distributed computing systems, synchronization algorithms are used to maintain synchronization.

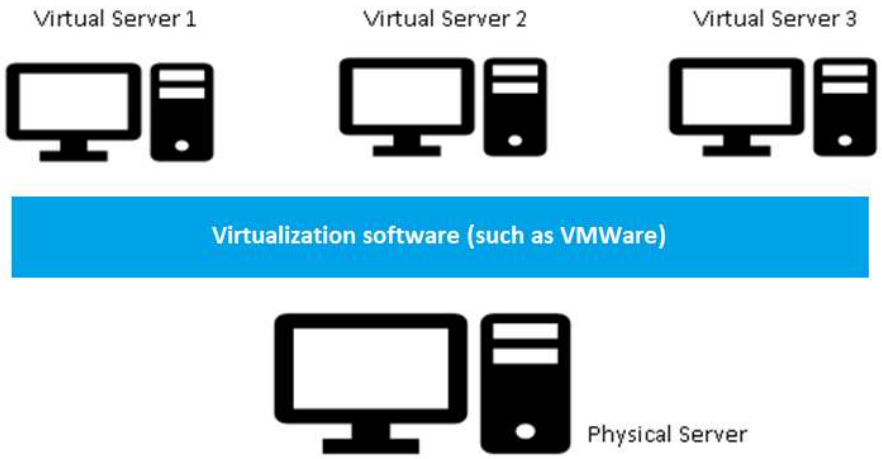
Parallel Computing vs. Distributed Computing

Usage

- Parallel computing is used to increase computer performance and for scientific computing.
  - High performance computing systems, Machine learning, Scientific modeling etc.
- Distributed computing is used to share resources and improve scalability.
  - Big data analytics, Distribute databases (such as NoSQL DBs) etc.

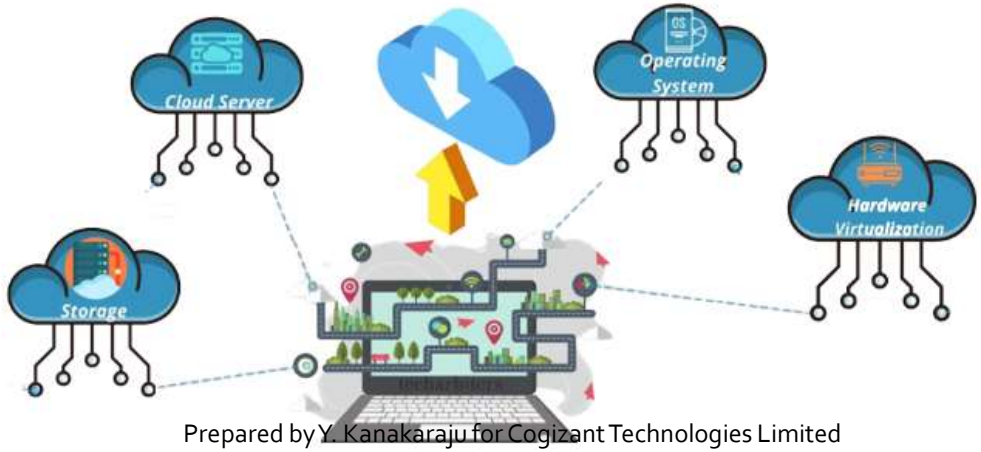
What is Virtualization?

Virtualization is a process that allows a computer to share its hardware resources with multiple digitally separated environments. Each virtualized environment runs within its allocated resources, such as memory, processing power, and storage. With virtualization, organizations can switch between different operating systems on the same server without rebooting.

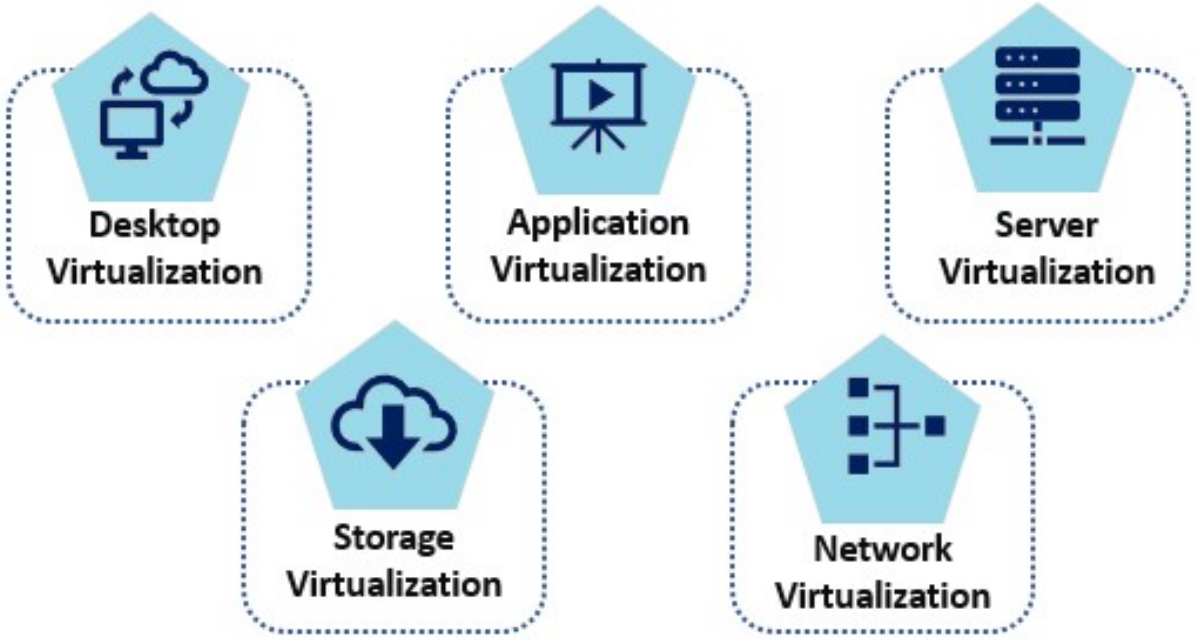


Why is virtualization important?

- By using virtualization, you can interact with any hardware resource with greater flexibility.
- Physical servers consume electricity, take up storage space, and need maintenance. You are often limited by physical proximity and network design if you want to access them. Virtualization removes all these limitations by abstracting physical hardware functionality into software. You can manage, maintain, and use your hardware infrastructure like an application on the web.



Types of virtualization

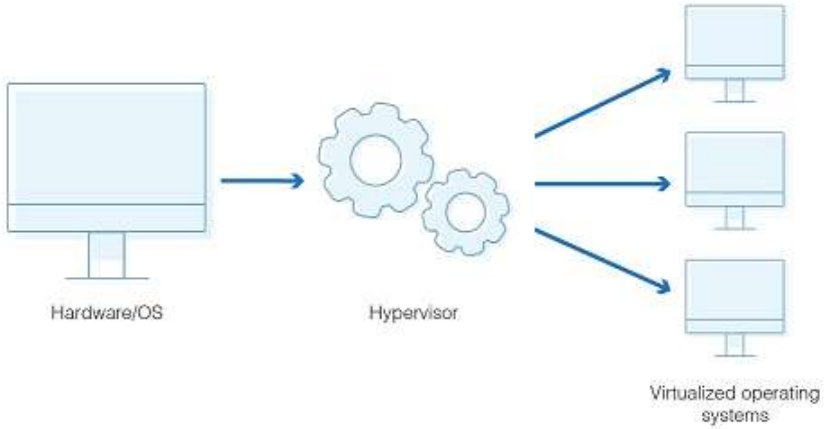


Virtual machine

- A virtual machine is a software-defined computer that runs on a physical computer with a separate operating system and computing resources.
- The physical computer is called the host machine and virtual machines are guest machines.
- Multiple virtual machines can run on a single physical machine. Virtual machines are abstracted from the computer hardware by a hypervisor.

Hypervisor

- The hypervisor is the virtualization software that you install on your physical machine.
- It is a software layer that acts as an intermediary between the virtual machines and the underlying hardware or host operating system.
- The hypervisor coordinates access to the physical environment so that several virtual machines have access to their own share of physical resources.

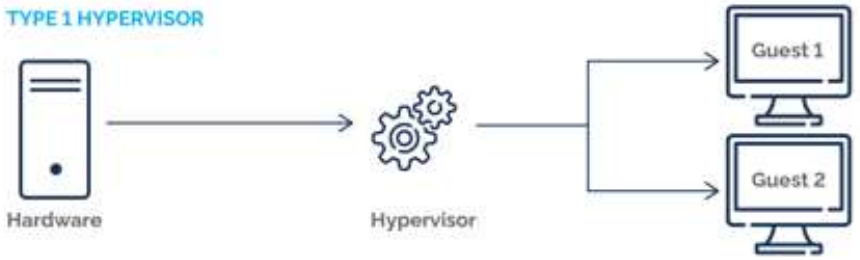


For example, if the virtual machine requires computing resources, such as computer processing power, the request first goes to the hypervisor. The hypervisor then passes the request to the underlying hardware, which performs the task.

Types of hypervisors

Type 1 hypervisors

A type 1 hypervisor—also called a bare-metal hypervisor—runs directly on the computer hardware. It has some operating system capabilities and is highly efficient because it interacts directly with the physical resources.



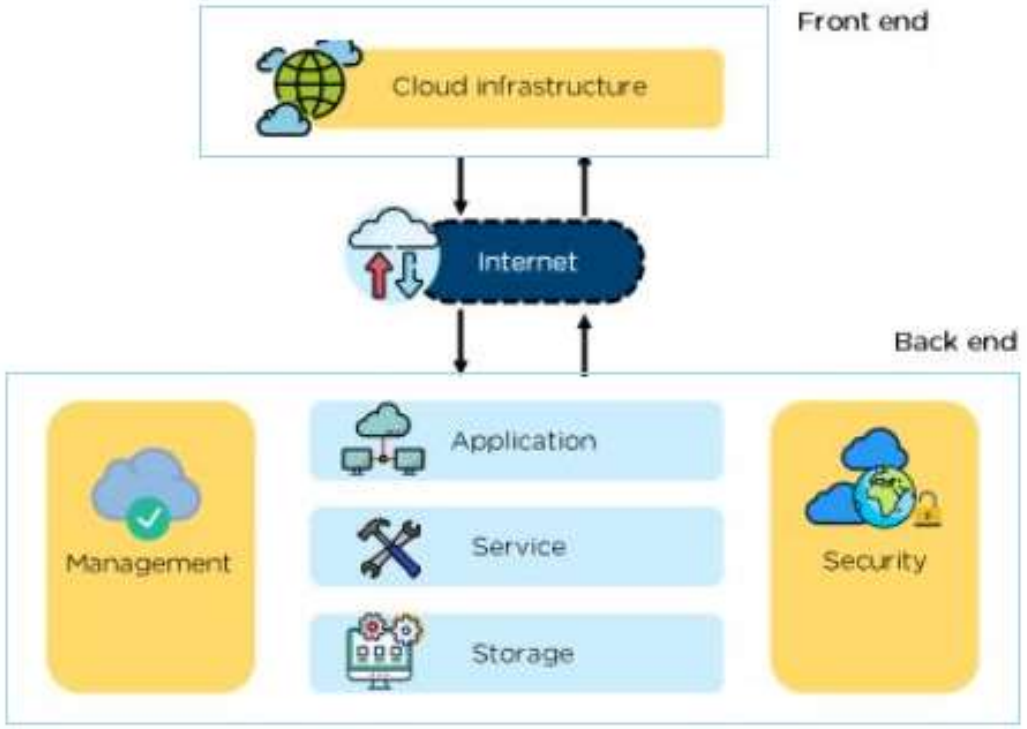
Type 2 hypervisors

A type 2 hypervisor runs as an application on computer hardware with an existing operating system. Use this type of hypervisor when running multiple operating systems on a single machine.

Cloud computing architecture

Cloud computing architecture is divided into three parts:

- Front-end platform
- Back-end platform
- Cloud-based delivery



Cloud computing architecture – Front-end

The front-end infrastructure of a cloud computing business platform is basically everything the end-user interacts with. It is the broader assimilation of various sub-components that together offer the user interface. This determines how the end-user connects to cloud computing as a whole.

Some important front-end components are:

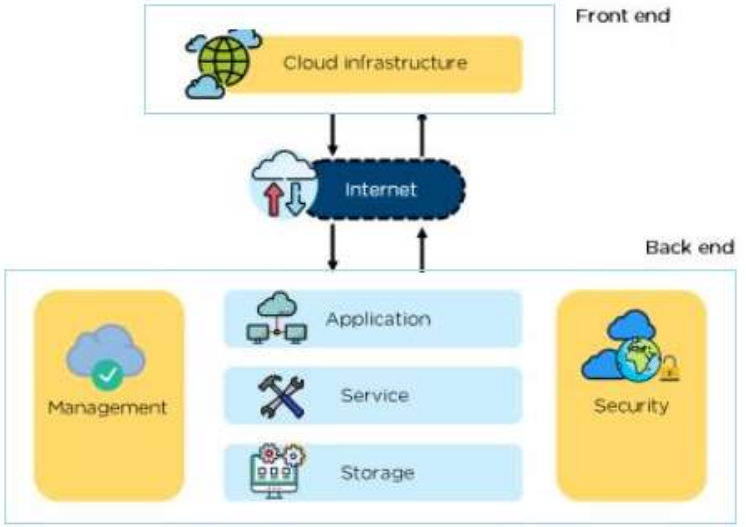
- **User Interface:** The user interface is essentially the interface that you experience each day. Cloud creates a seamless environment where end-users can complete tasks without ever needing to open up any software on their local machines.
- **Software:** The software architecture in the front end is what runs on the user's end such as client-side applications or browsers which are in charge of presenting data to users.
- **Client Device or Network:** The hardware at the end user's side is referred to as the client-side device.

Cloud computing architecture – Back-end

Back-end components are responsible for monitoring all the programs that run the application on the front-end. The back-end architecture in the cloud empowers the front-end architecture. It is comprised of hardware & storage that are located on a remote server. The cloud service provider handles and controls this backend architecture.

Some important components in the back-end are:

- Application
- Service
- Storage
- Management
- Security
- Infrastructure



Cloud delivery models

A cloud delivery model refers to any cloud-based solution one can access through a web browser from any device with internet capabilities.

- **Software as a Service (SaaS)** - This is the software distribution model whereby developers put their applications into a cloud-based delivery system. Customers access these applications via the Internet, usually through a browser. Ex: Google Workspace, Dropbox, Salesforce, Cisco WebEx etc.

- **Platform as a Service (PaaS)** - This type of model is often run by an organization where the users can not only create and run applications on the cloud but also effectively maintain them themselves. Ex: Google App Engine, Apache Stratos, Heroku, Force.com etc.

- **Infrastructure as a Service (IaaS)** - This model provides the infrastructure necessary for companies to run their operations. It includes virtual and non-virtual servers, storage, and data center space all in one place.

AWS Basics

What is AWS?

Amazon Web Services (AWS) is a comprehensive cloud computing platform that includes:

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)

AWS services offer scalable solutions for :

- Compute
- Storage
- Databases
- Analytics *and many more…*

Why AWS ?

- **Easy to use**

  AWS is designed to allow application providers to quickly and securely host your applications – whether an existing application or a new SaaS-based application. You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.

- **Flexible**

  AWS enables you to select the operating system, programming language, web application platform, database, and other services you need. With AWS, you receive a virtual environment that lets you load the software and services your application requires. This eases the migration process for existing applications while preserving options for building new solutions.

- **Cost-Effective**

  You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments. For more information on comparing the costs of other hosting alternatives with AWS, see the AWS Economics Center.

Why AWS ?

- **Reliable**

  With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon.com's multi-billion dollar online business that has been honed for over a decade.

- **Scalable and high-performance**

  Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

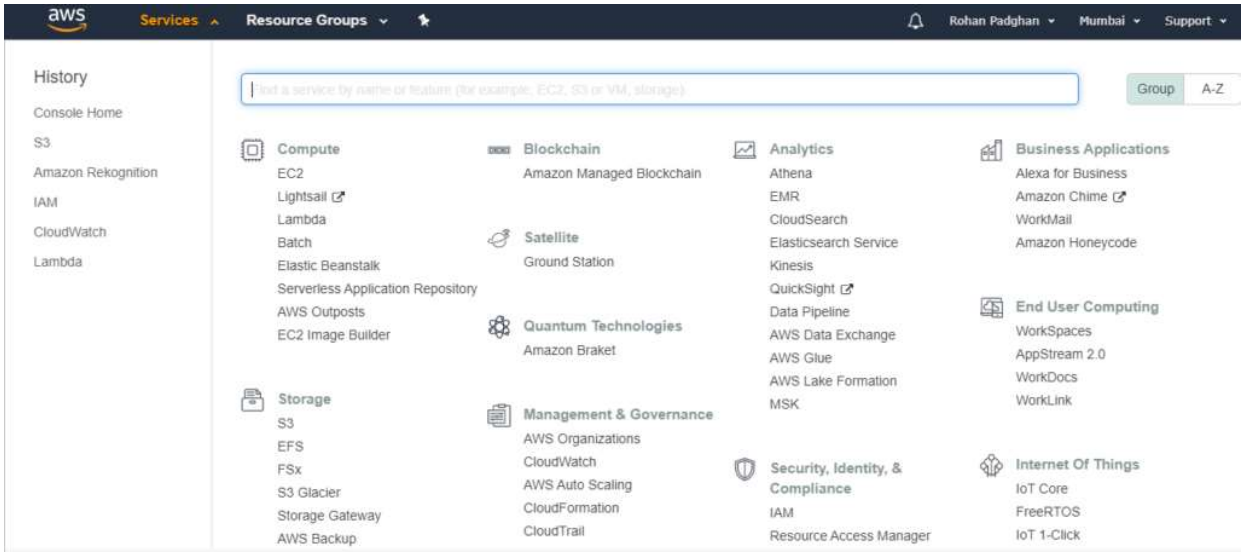
- **Secure.**

  AWS utilizes an end-to-end approach to secure and harden our infrastructure, including physical, operational, and software measures. For more information, see the AWS Security Center.

Understanding console and various services

Demo.

The AWS Management Console is a web application that comprises and refers to a broad collection of service consoles for managing AWS resources. When you first sign in, you see the console home page.



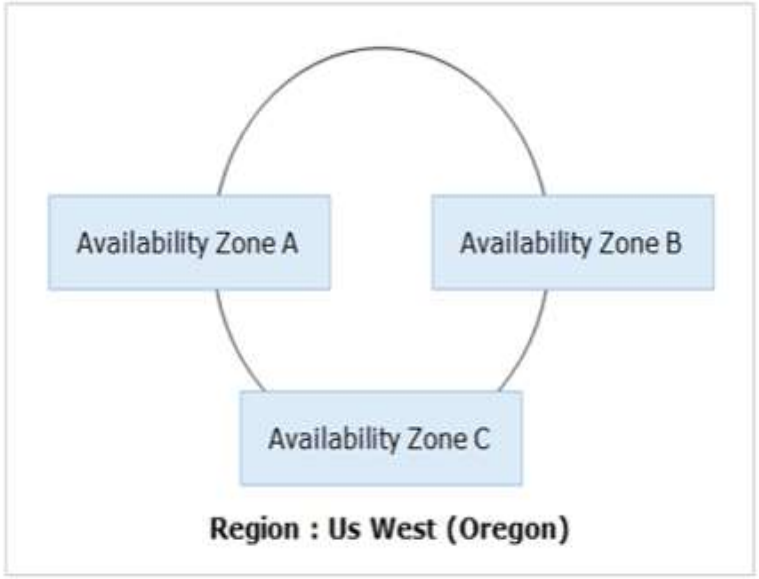
AWS Global Cloud Infrastructure

Global cloud infrastructure of AWS

The AWS Cloud spans 99 Availability Zones within 31 geographic regions around the world, with announced plans for 15 more Availability Zones and 5 more AWS Regions in Canada, Israel, Malaysia, New Zealand, and Thailand. (as of March 2023)

Regions & Availability zones

- AWS **Region** is a physical location around the world where AWS clusters data centers called **Availability Zones**. Each group of logical data centers is called an Availability Zone (AZ).
- Each AWS Region consists of multiple, isolated, and physically separate Availability Zones within a geographic area.
- Each AZ has independent power, cooling, and physical security and is connected via redundant, ultra-low-latency networks.



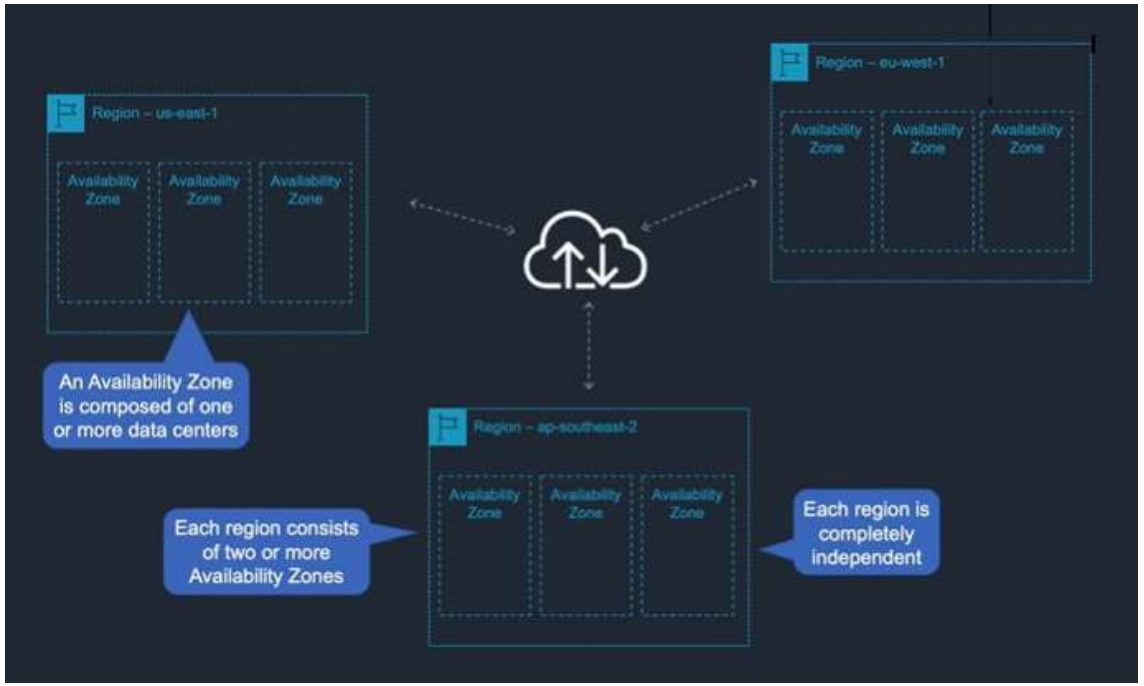
Regions & Availability zones

Region – Two or more AZs

Availability Zone – One or more data centers

Each region is completely independent

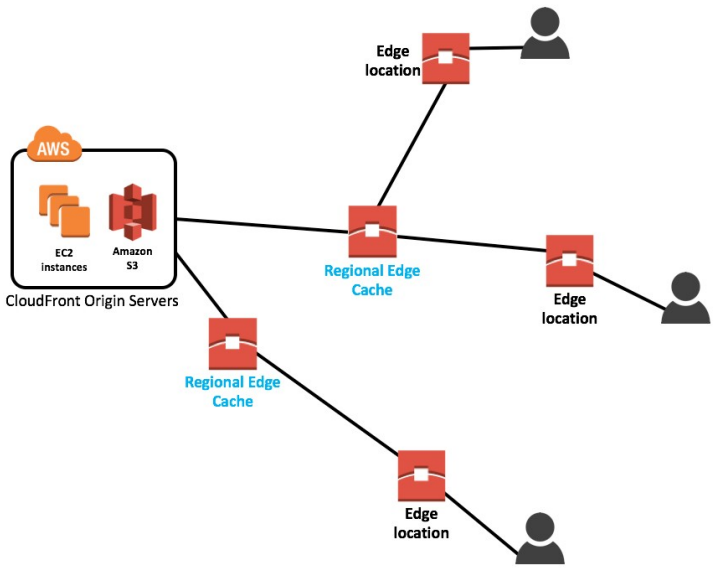
All regions are connected via high bandwidth, fully redundant network.



AWS Edge Networks

The other component of the AWS Global Cloud Infrastructure is the edge networks of Point-of-Presence or PoP. It consists of **Edge Locations** and **Regional Edge Caches,** which enables you to distribute your content with low-latency to your global users. Basically, a PoP serves as an access point that allows two different networks to communicate with each other.

By using these global edge networks, a user request doesn't need to travel far back to your origin just to fetch data. The cached contents can quickly be retrieved from regional edge caches that are closer to your end-users.



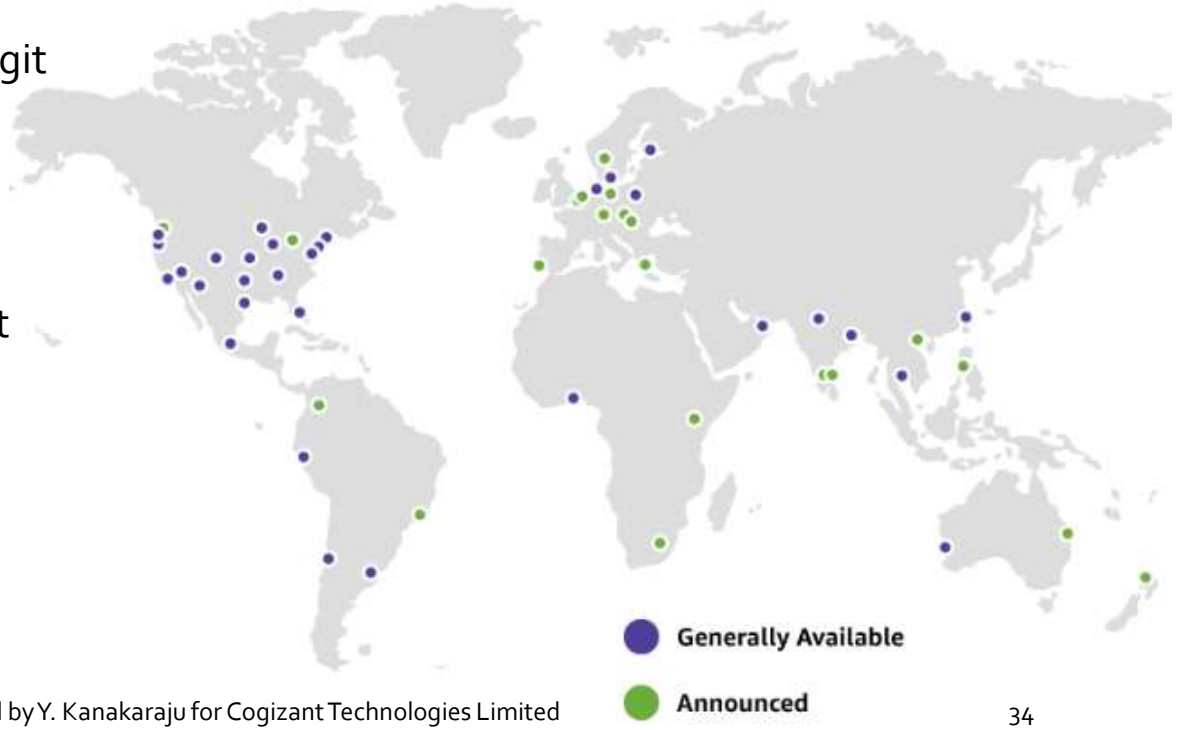
AWS provides a service called **Amazon CloudFront** to leverage edge network for low-latest content delivery.

AWS Local Zones

AWS Local Zones are a type of infrastructure deployment that places compute, storage, database, and other select AWS services close to large population and industry centers.

Local zone gives users access to single-digit milliseconds latency with AWS Direct Connect and the ability to meet data redundancy requirements.

Local zones are connected to their parent region via AWS' redundant and high bandwidth private network.



AWS Wavelength Zones

Wavelength Zones are AWS infrastructure deployments that embed AWS compute and storage services within telecommunications providers' data centers at the edge of the 5G network, so application traffic can reach application servers running in Wavelength Zones without leaving the mobile providers' network.

Wavelength Zones by region:

https://docs.aws.amazon.com/wavelength/latest/developerguide/available-wavelength-zones.html

AWS Outposts

AWS Outposts is a family of fully managed solutions delivering AWS infrastructure and services to virtually any on-premises or edge location for a truly consistent hybrid experience.

Outposts solutions allow you to extend and run native AWS services on premises, and is available in a variety of form factors, from 1U and 2U Outposts servers to 42U Outposts racks, and multiple rack deployments.

With AWS Outposts, you can run some AWS services locally and connect to a broad range of services available in the local AWS Region.

The 'U' in any server description is short for "RU" which stands for Rack Unit -- this is the standardized designation for the form factor of the server: 1U = 1.75" in height or thickness. 2U is 1.75" x2 = 3.5 inches.

AWS Identity and Access Management (IAM)

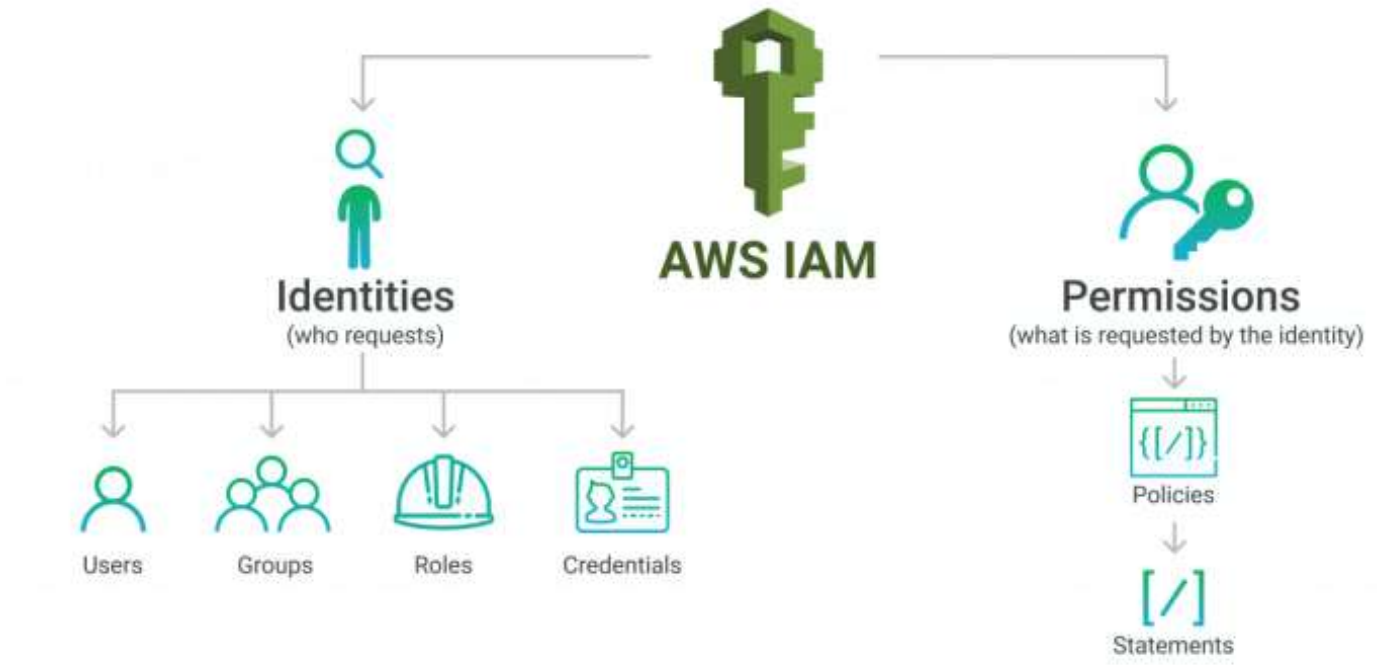
AWS IAM

- AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources for your users.
- IAM is used to control
  - **Identity** – who can use your AWS resources (authentication)
  - **Access** – what resources they can use and in what ways (authorization)

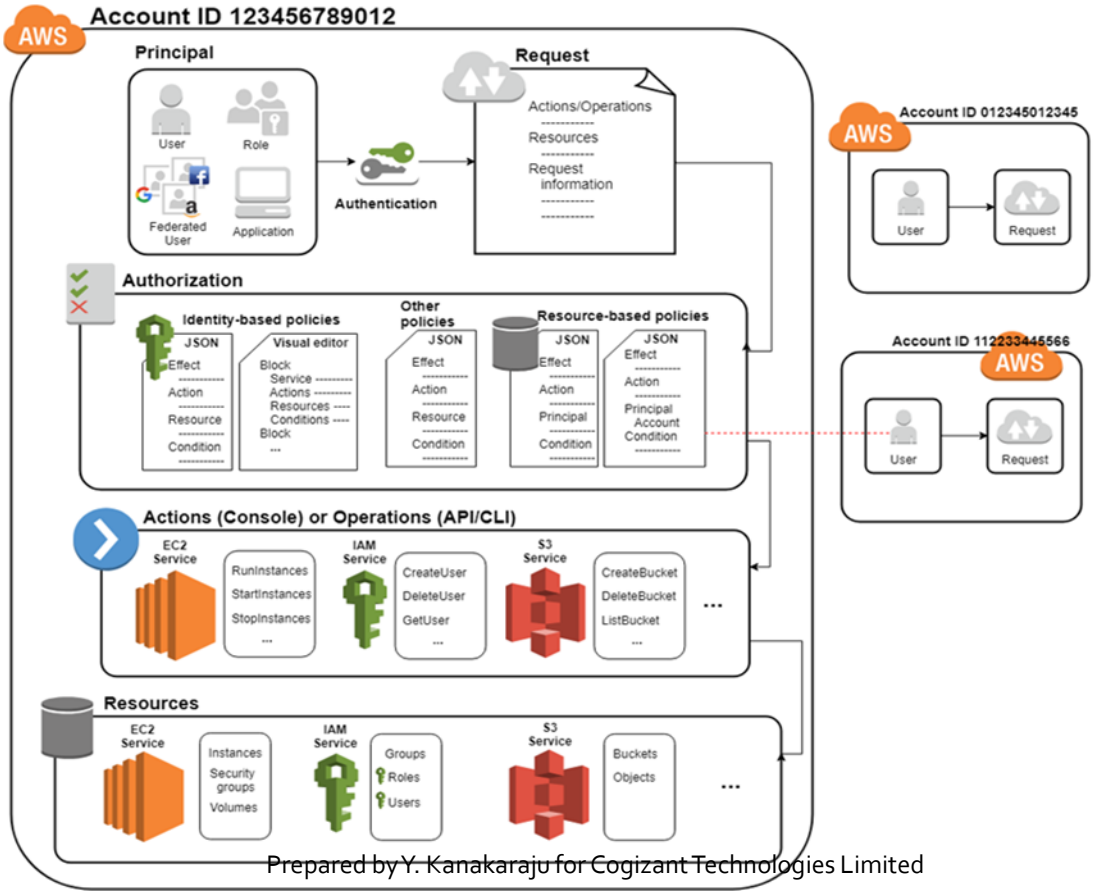
AWS IAM

With AWS Identity and Access Management (IAM), you can specify who or what can access services and resources in AWS, centrally manage fine-grained permissions, and analyze access to refine permissions across AWS.

AWS IAM


AWS IAM
Identities
(who requests)
Users
Groups
Roles
Credentials
Permissions
(what is requested by the identity)
Policies
[/]
Statements

How IAM works ?



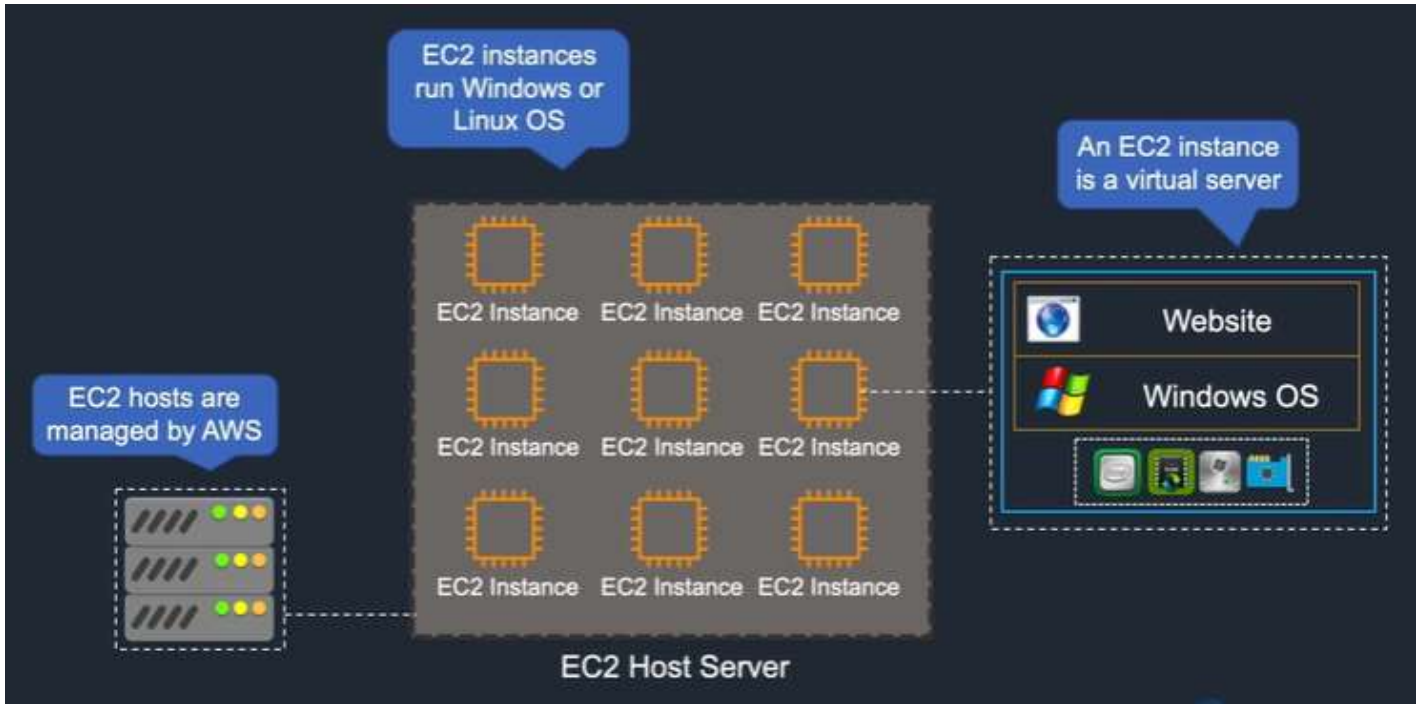
Amazon Elastic Compute Cloud (EC2)

Amazon EC2

- Amazon EC2 is a web service that provides resizable compute capacity in the cloud.
- Amazon EC2 reduces the time required to obtain and boot new user instances by providing virtual machines in the cloud in minutes.
- You can scale the compute capacity up and down as per the computing requirement changes.
- Amazon EC2 allows you to pay for the capacity that you actually use.
- Amazon EC2 provides the developers with the tools to build resilient applications that isolate themselves from some common scenarios.

An EC2 instance is a virtual server on the cloud.

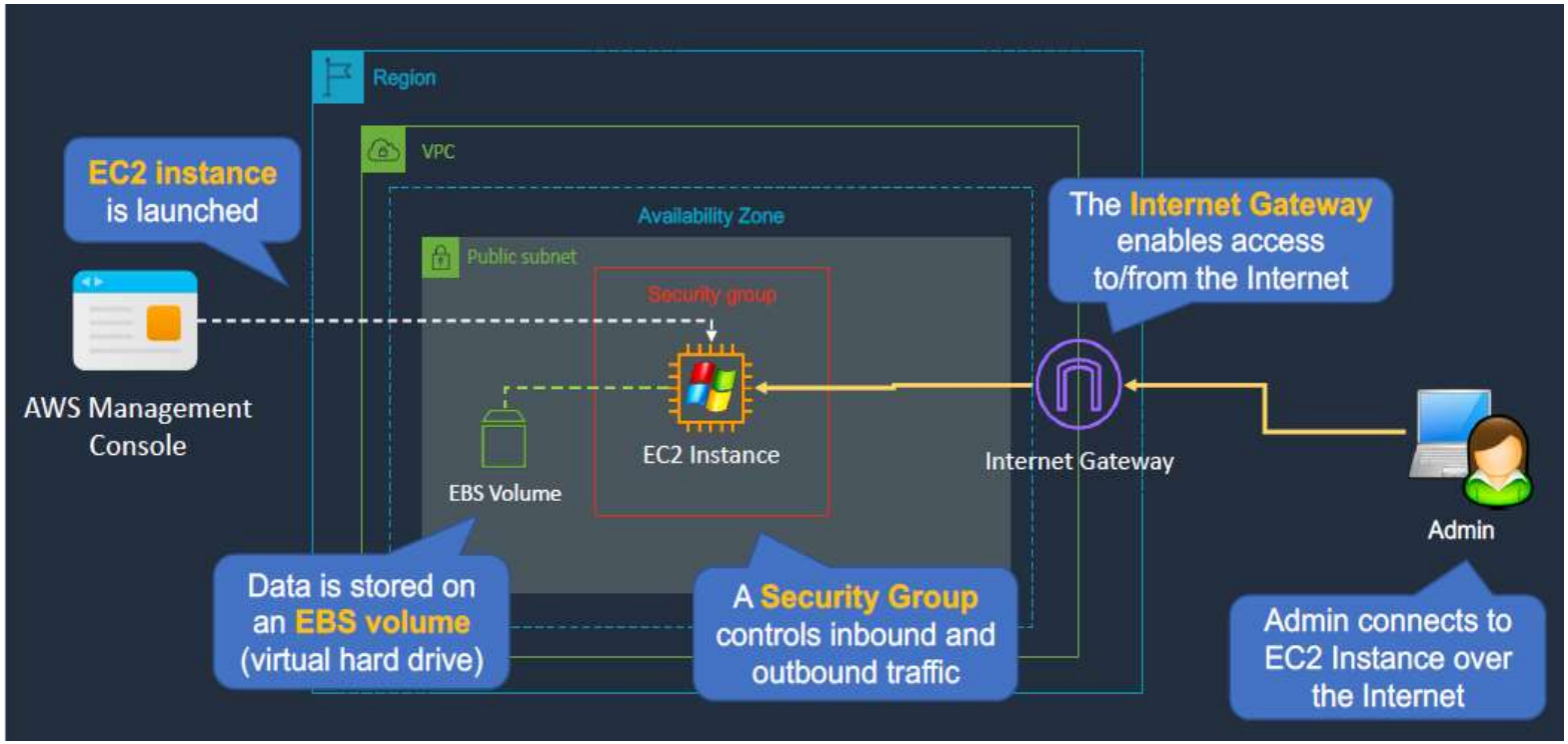
Amazon EC2


EC2 instances run Windows or Linux OS
An EC2 instance is a virtual server
EC2 Instance
EC2 Instance
EC2 Instance
EC2 Instance
EC2 Instance
EC2 Instance
EC2 Instance
EC2 Instance
EC2 Instance
Website
Windows OS
EC2 hosts are managed by AWS
EC2 Host Server

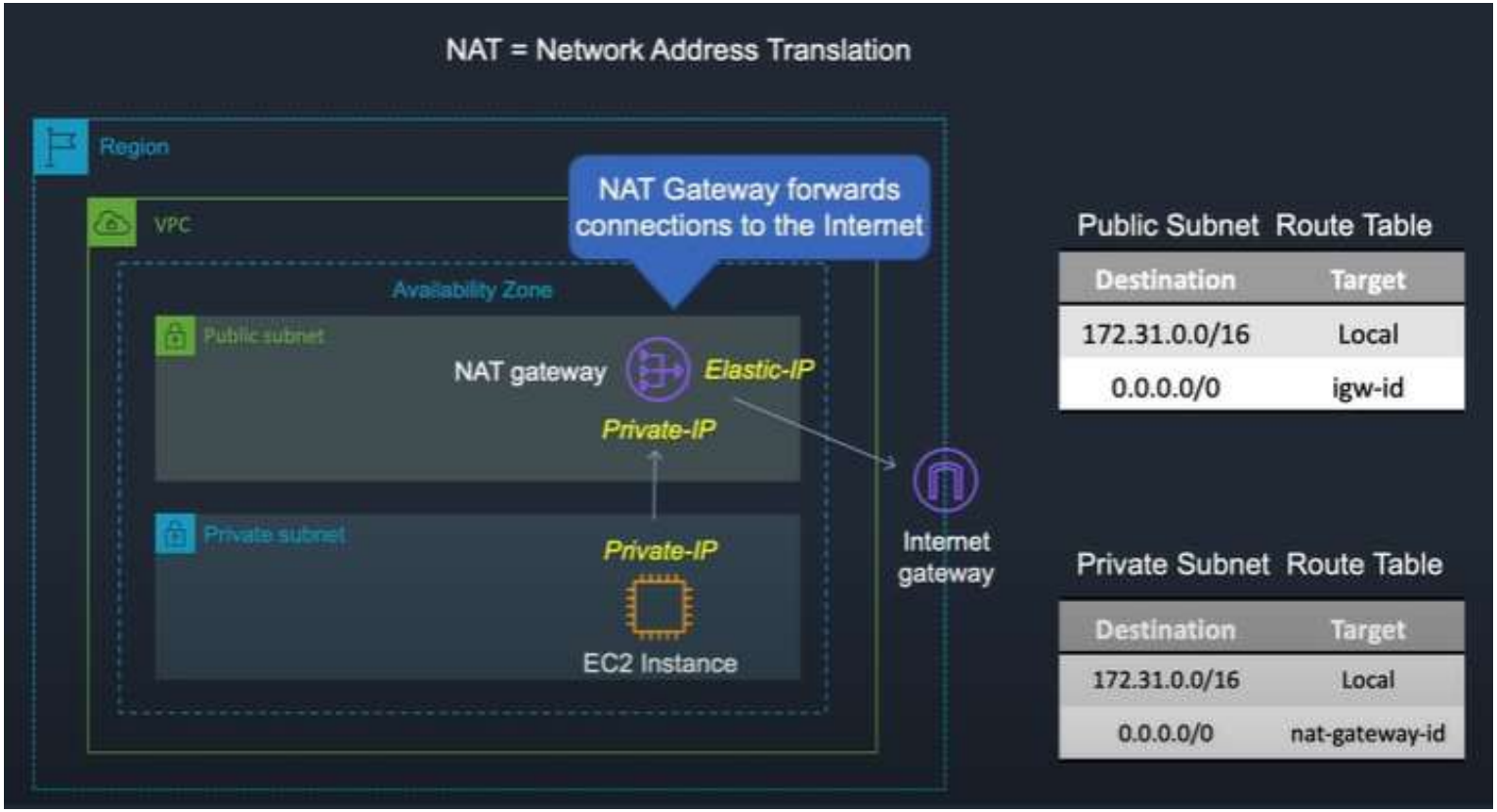
IP address for EC2 instances

- **Public IP address**
  - Lost when instance is stopped
  - Used in public subnets
  - No charge
  - Associated with a private address on the instance
  - Can not be moved between instances
- **Private IP address**
  - Retained when the instance is stopped
  - Used in public and private subnets
- **Elastic IP address**
  - Static public IP address
  - Charged if not used
  - Associated with a private address on the instance
  - Can be moved between instances and elastic network adapters

Amazon EC2 in a public subnet



Amazon EC2 in a public & private subnets



Amazon EC2 in a public & private subnets

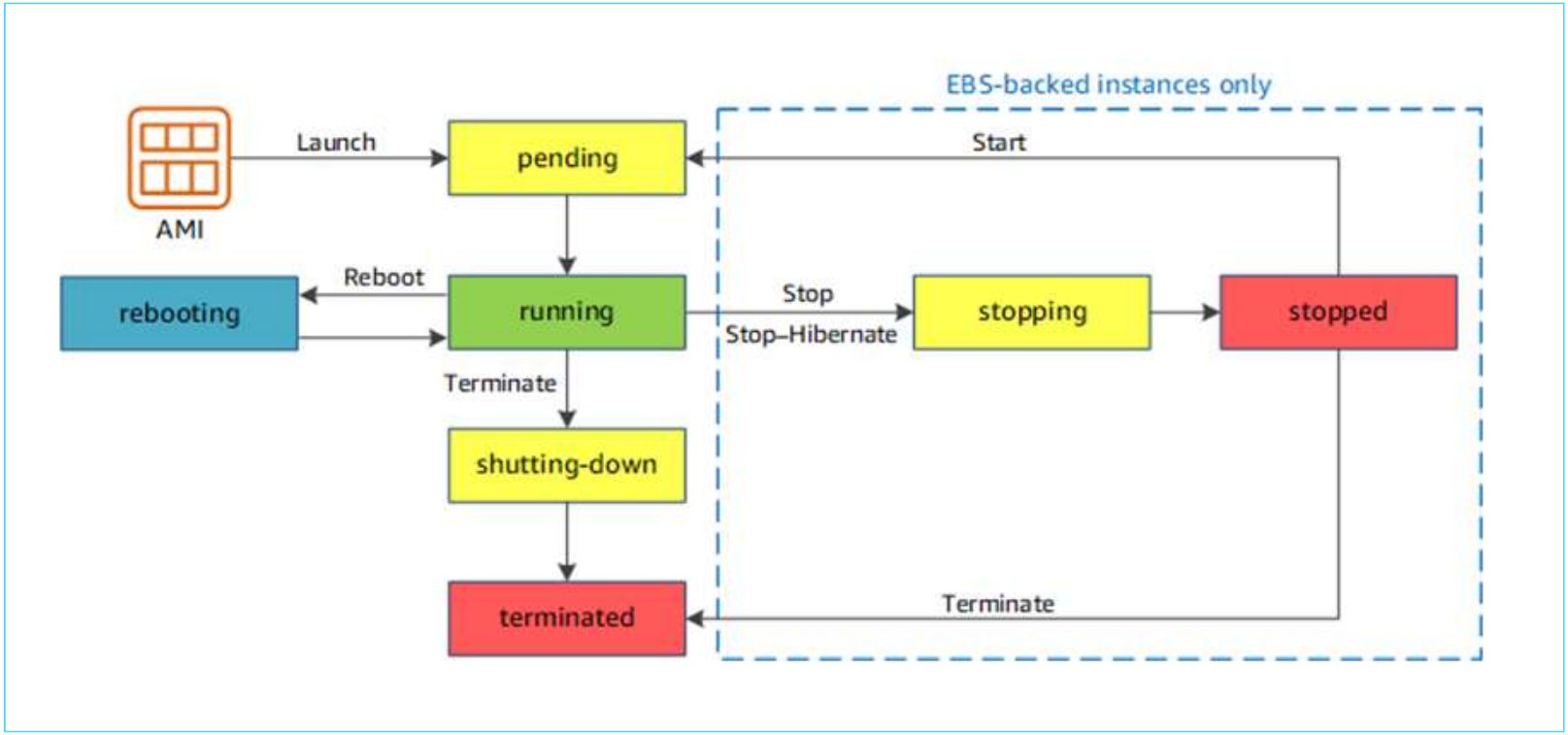
- An EC2 instance in the public subnet has a public (non-static) IP address and can connect to the internet via a **Public subnet route table**.
- An EC2 instance in the private subnet can only connect to other EC2 instances with the same subnet.
- If you want to connect an EC2 instance in the private subnet to the internet, you have to use **Private subnet route table** and implement a **NAT Gateway for an EC2 node in the public subnet.** The private EC2 instance can then connect to internet via the NAT gateway of the node in the public subnet.

Amazon EC2 instance types

Family	Type	vCPUs	Memory (GiB)
General Purpose	t2.micro	1	1
Compute Optimized	c5n.large	2	5.25
Storage Optimized	d2.xlarge	4	30.5
Memory Optimized	r5ad.large	2	16
Accelerated Computing Instances	g2.2xlarge	8	15

URL: https://www.amazonaws.cn/en/ec2/instance-types/

EC2 instance life cycle



EC2 instance life cycle

Instance state	Description	Instance usage billing
pending	The instance is preparing to enter the running state. An instance enters the pending state when it launches for the first time, or when it is started after being in the stopped state.	Not billed
running	The instance is running and ready for use.	Billed
stopping	The instance is preparing to be stopped or stop-hibernated.	Not billed if preparing to stop; Billed if preparing to hibernate
stopped	The instance is shut down and cannot be used. The instance can be started at any time.	Not billed
shutting-down	The instance is preparing to be terminated.	Not billed
terminated	The instance has been permanently deleted and cannot be started.	Not billed

EC2 instance states - stopping

- Applies to EBS backed instances only
- No charge for stopped instances
- EBS volumes remain attached (chargeable)
- Data in RAM is lost
- Instance is migrated to a different host
- Private IPv4 addresses and IPv6 addresses retained; public IPv4 addresses released
- Associated Elastic IPs retained

EC2 instance states - hibernating

- Applies to on-demand or reserved Linux instances
- Contents of RAM saved to EBS volume
- Must be enabled for hibernation when launched
- Specific prerequisites apply
- When started (after hibernation):
  - The EBS root volume is restored to its previous state
  - The RAM contents are reloaded
  - The processes that were previously running on the instance are resumed
  - Previously attached data volumes are reattached and the instance retains its instance ID

EC2 instance states - rebooting

- Equivalent to an OS reboot
- DNS name and all IPv4 and IPv6 addresses retained
- Does not affect billing

EC2 instance states - retiring

- Instances may be retired if AWS detects irreparable failure of the underlying hardware that hosts the instance
- When an instance reaches its scheduled retirement date, it is stopped or terminated by AWS

EC2 instance states - terminating

- Means deleting the EC2 instance
- Cannot recover a terminated instance
- By default root EBS volumes are deleted

EC2 instance states - recovering

- CloudWatch can be used to monitor system status checks and recover instance if needed
- Applies if the instance becomes impaired due to underlying hardware / platform issues
- Recovered instance is identical to original instance

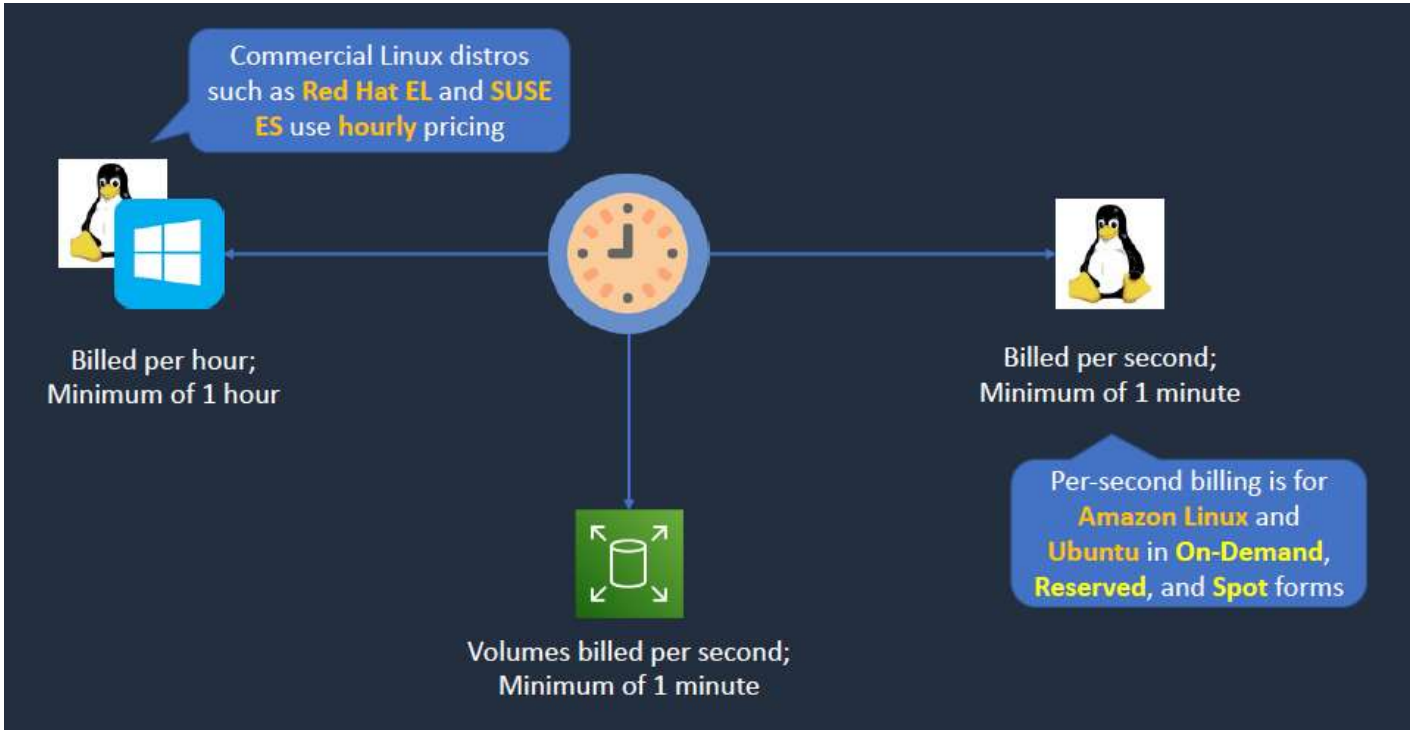
EC2 pricing options

- **On-Demand**
  - Standard rate - no discount
  - No commitments
  - Use for development/testing, short-term or unpredictable workloads
- **Reserved**
  - 1 or 3-year commitment
  - Up to 75% discount
  - Use for steady-state, predictable workloads and reserved capacity
- **Spot-Instances**
  - Bid for unused capacity
  - Up to 90% discount;
  - Can be terminated at any time
  - Use for Workloads with flexible start and end times

EC2 pricing options

- **Dedicated Instances**
  - Physical isolation at the host hardware level from instances belonging to other customers
  - Pay per instance
- **Dedicated Hosts**
  - Physical server dedicated for your use
  - Socket/core visibility, host affinity
  - Pay per host
  - Workloads with server-bound software licenses
- **Savings Plans**
  - Commitment to a consistent amount of usage (EC2 + Fargate + Lambda)
  - Pay by $/hour
  - 1 or 3-year commitment

EC2 billing


Commercial Linux distros such as Red Hat EL and SUSE ES use hourly pricing
Billed per hour;
Minimum of 1 hour
Billed per second;
Minimum of 1 minute
Per-second billing is for Amazon Linux and Ubuntu in On-Demand, Reserved, and Spot forms
Volumes billed per second;
Minimum of 1 minute

Amazon EC2 Labs

Lab 1: Create a Key-Pair

Instructions Document: L01-Create-KeyPair.txt

- A key-pair is used to authenticate while launching an EC2 instance.
- A key-pair consists of a private and a public key.
- When you create a key-pair, a random private key and the associated public key will be created.
- Private Key files will be provided to you for download.
  - We use a **.pem** file for Linux/Mac and **.ppk** file for windows via puTTY.
- Public Key will be managed somewhere inside AWS repositories.
- When you launch an ec2 instance with a specific key pair, the public key will be added to authorized keys on the server under standard user.

EC2 Standard user names

- **ec2-user** : for 'Amazon Linux' instances
- **ubuntu** : for 'Ubuntu' instances

Lab 2: Launch an EC2 instance

Instructions Document: L02-Lauch-EC2-instance.txt

While launching an EC2 instance, we set some of these important options:

1. Application and OS images
   - Here we select the OS (such as Amazon Linux, macOS, Ubuntu, Windows etc.) and a specific AMI with in that OS.

2. Network settings
   - Here we create or select a security group for the instance

3. Storage settings
   - Here we attach root EBS volume, any other additional EBS volumes and file systems.

Lab 3: Connect to EC2 instance using ‘Instance connect’

Instructions Document: L03- EC2-InstanceConnect.txt

Amazon EC2 Instance Connect provides a simple and secure way to connect to your Linux instances using Secure Shell (SSH). With EC2 Instance Connect, you use AWS Identity and Access Management (IAM) policies and principals to control SSH access to your instances, removing the need to share and manage SSH keys. All connection requests using EC2 Instance Connect are logged to AWS CloudTrail so that you can audit connection requests.

Lab 4: Connect to Linux EC2 instance using SSH

Instructions Document: L04-EC2-SSH.txt

We can connect to Linux EC2 instance using SSH in two different methods:

1. We can use SSH command and the .pem file (the private key file) of the key-value pair by running a command in a Linux terminal that looks something as shown below:

   ```
   ssh -i ~/.ssh/mykey.pem ec2-user@ec2-3-80-227-2.compute-1.amazonaws.com
   ```

2. We can use PuTTY to connect to EC2 instance as well. To use PuTTY tool we need a **.ppk file**. This file can either be downloaded when creating the key-pair itself, or can be generated from the .pem file using PuTTYgen tool.

Lab 5: Connect to Windows EC2 instance using RDP

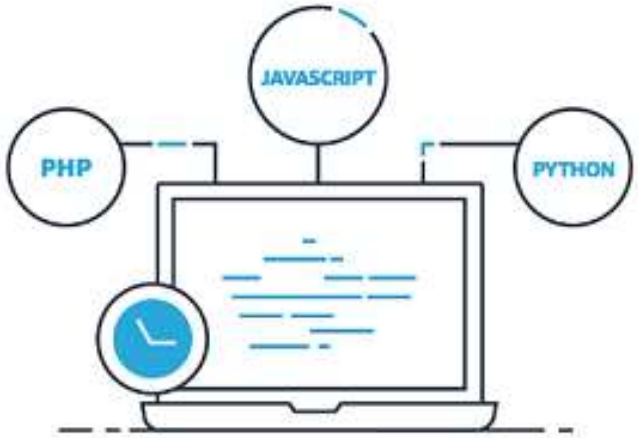
Instructions Document: L03-EC2-Windows-RDP.txt

In this lab we launch a Windows EC2 instance and connect to that instance from a client machine using Remote desktop application.

AWS Cloud9

AWS Cloud9

- AWS Cloud9 is a **cloud-based IDE** that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal.
- Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects.



- Since your Cloud9 IDE is cloud-based, you can work on your projects from your office, home, or anywhere using an internet-connected machine.
- With Cloud9, you can quickly share your development environment with your team, enabling you to pair program and track each other's inputs in real time.

Lab 1: Setup and work in a Cloud9 environment

Instructions Document: L01-Cloud9.txt

Amazon Simple Storage Service (S3)

Amazon S3

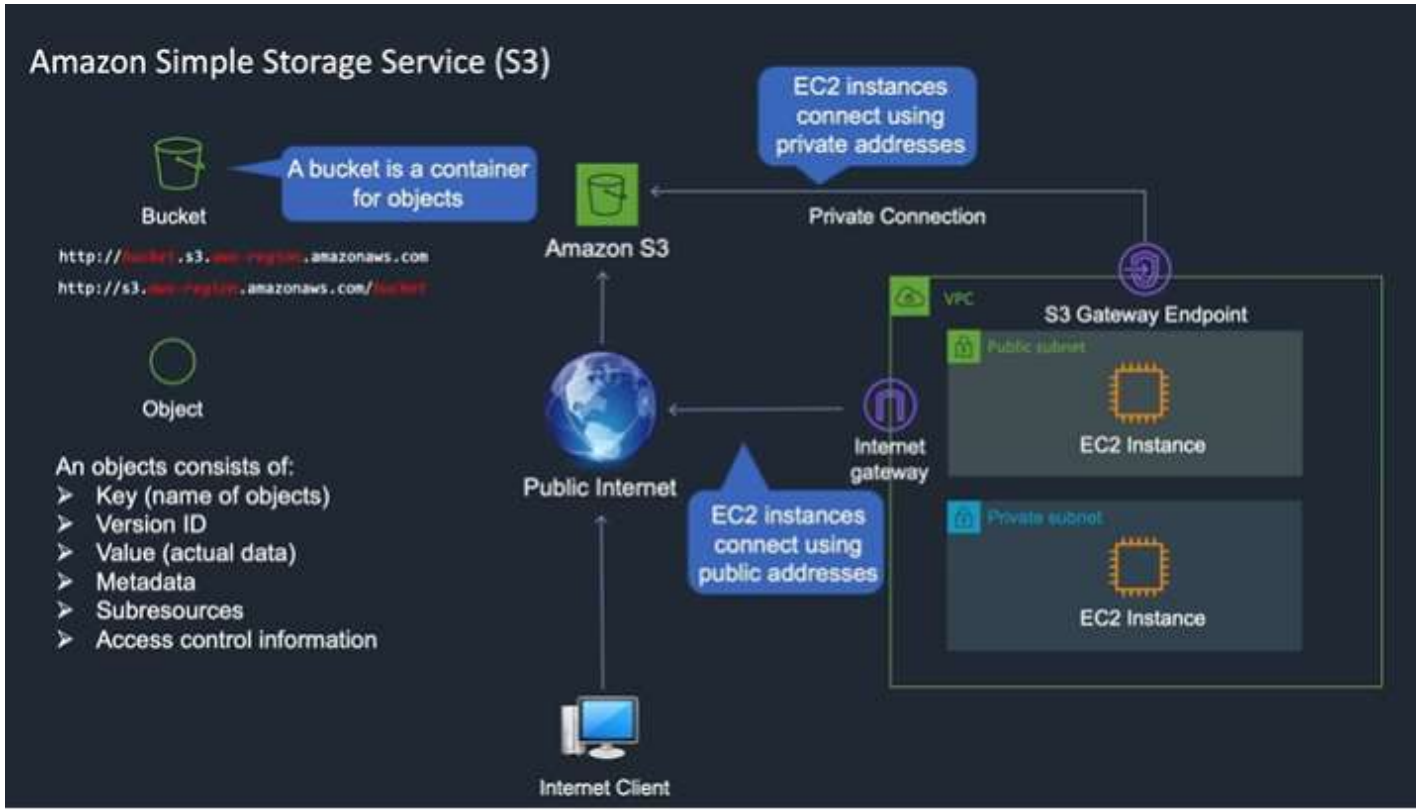
- Amazon **Simple Storage Service** (Amazon **S3**) is an **object storage service** that offers industry-leading scalability, data availability, security, and performance.

- S3 is a web-based service designed for online backup and archiving of data and application programs. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere via an internet URL.

- S3 allows to upload, store, and download any type of files up to **5 TB** in size.

- In S3 we store **objects** in **buckets**
  - Bucket is a container of objects. Each bucket can have any number of objects.
  - Bucket name must be globally unique (as it is accessed via a URL).
  - Bucket is created in a region.

Amazon S3


Amazon Simple Storage Service (S3)
Bucket
A bucket is a container for objects
http://bucket.s3.web-region.amazonaws.com
http://s3.web-region.amazonaws.com/bucket
Object
An objects consists of:
Key (name of objects)
Version ID
Value (actual data)
Metadata
Subresources
Access control information
Amazon S3
EC2 instances connect using private addresses
Private Connection
VPC
S3 Gateway Endpoint
Public subnet
EC2 Instance
Internet gateway
Public Internet
EC2 instances connect using public addresses
Private subnet
EC2 Instance
Internet Client

Amazon S3

- You can store any type of file in S3
- Files can be anywhere from 0 bytes to 5 TB
- There is unlimited storage available
- S3 is a universal namespace so bucket names must be unique globally. However, you create your buckets within a region
- It is a best practice to create buckets in regions that are physically closest to your users to reduce latency
- There is no hierarchy for objects within a bucket
- Delivers strong read-after-write consistency

Amazon S3 Buckets

- Files are stored in buckets
- A bucket can be viewed as a container for objects
- A bucket is a flat container of objects
- It does not provide a hierarchy of objects
- You can use an object key name (prefix) to mimic folders
- 100 buckets per account by default
- You can store unlimited objects in your buckets
- You cannot create nested buckets

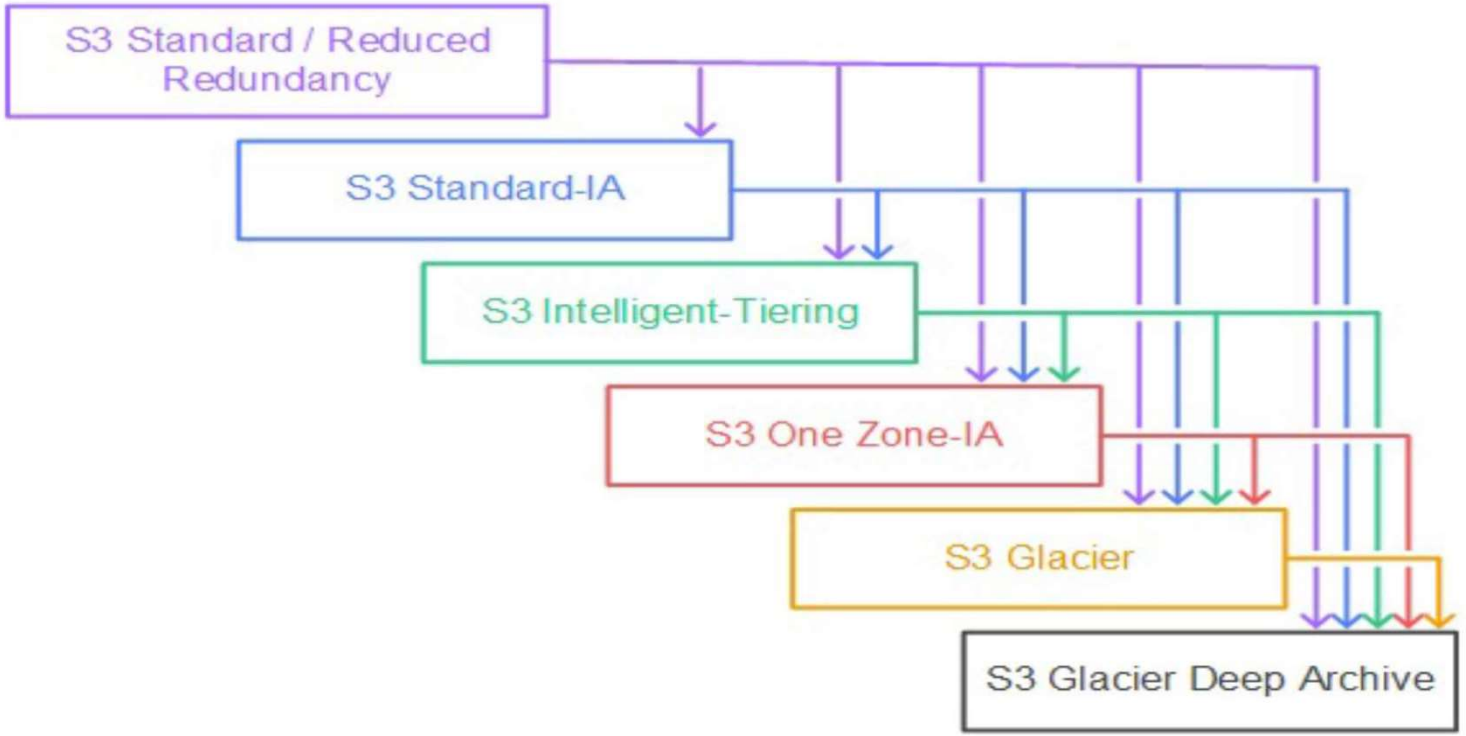
Amazon S3 Objects

- An object is a file uploaded to S3
- S3 supports any file type
- Each object is stored and retrieved by a unique key
- Objects remain in the region they are stored you setup replication
- Permissions can be defined on objects at any time
- Storage class is set at the object level

Amazon S3 storage classes

	S3 Standard	S3 Intelligent Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999%	99.999999999%	99.999999999%	99.999999999%	99.999999999%	99.999999999%
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

Amazon S3 storage class transitions



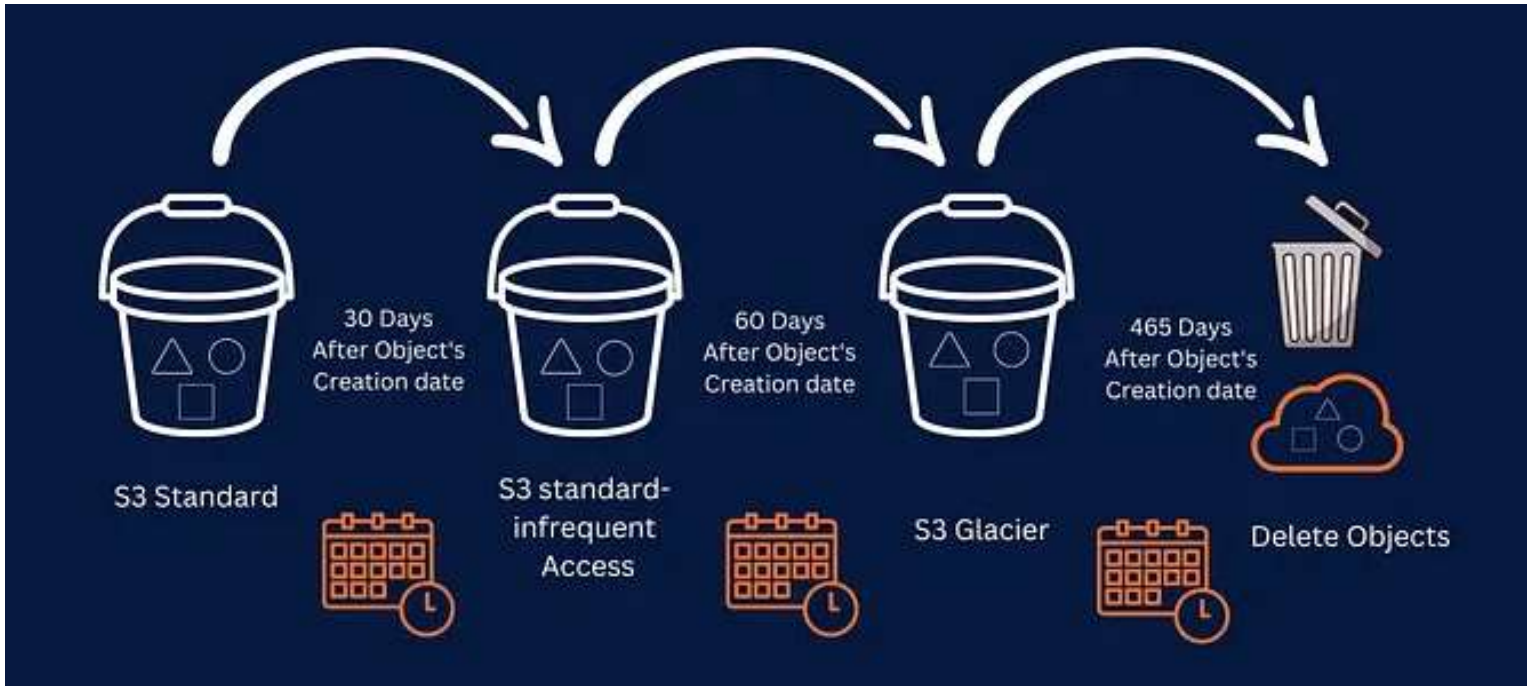
Amazon S3 versioning

- In S3, when you create a bucket, versioning is disabled by default.
- To enable versioning:
  - Go to properties tab of the bucket
  - Click on **Edit** button in the **Bucket versioning**
  - Enable the versioning
- If versioning is enabled, when you upload the same object multiple times to the same bucket, different versions are create.
- When you delete an object, a delete marker is created on the object. You can delete some of the delete marked files to go back to older versions.
- To delete an object from a bucket, you should not only delete the object itself, but also delete all the versions as well.

Amazon S3 life cycle management

- Lifecycle Management is used to store objects cost-effectively throughout their lifecycle. A lifecycle configuration is a set of rules that define the actions applied by S3 to a group of objects.
- The lifecycle defines two types of actions:
  - **Transition actions**: define when objects transition to another storage class.
    - For example, you choose to transit the objects to the Standard IA storage class 30 days after you have created them or archive the objects to the Glacier storage class 60 days after you have created them.
  - **Expiration actions**: define when objects expire. Amazon S3 deletes the expired object on your behalf.

Amazon S3 life cycle management



Amazon S3 server access logging (SAL)

- Disabled by default
- If enabled, SAL provides detailed records for the requests that are made to a bucket
  - Details include the requester, bucket name, request time, request action, response status, and error code (if applicable)
- Only pay for the storage space used
- Must configure a separate bucket as the destination (can specify a prefix)
- Must grant write permissions to the Amazon S3 Log Delivery group on destination bucket

Amazon S3 bucket policy

- Bucket policies are an IAM mechanism for controlling access to resources. They are a critical element in securing your S3 buckets against unauthorized access and attacks.
- A bucket policy is a JSON object that is attached to a bucket and allows you to manage access to specific S3 objects in a bucket.
- You can specify permissions for each resource to allow or deny actions requested by a principal (a user or role).
- When you create a new Amazon S3 bucket, you should set a policy granting the relevant permissions to the data forwarder's principal roles.

Amazon S3 Labs

Lab 1: S3 bucket with versioning & storage class

Instructions Document: L01-S3-Versioning-Storageclass.txt

In this lab we demo:

1. Creating and using S3 buckets
2. Applying versioning to S3 buckets
3. Applying & changing storage classes for an S3 bucket
4. Defining life cycle rules for an S3 bucket

Lab 2: S3 user policies and bucket policies

Instructions Document: L02-S3-PermissionPolicies.txt

In this lab we demo:

1. User policies: Apply inline policies to a specific user to control what kind of permission he may have one or more S3 buckets

2. Bucket policies: Apply bucket policy to a bucket to control which IAM user/role can access this bucket.

AWS Lambda

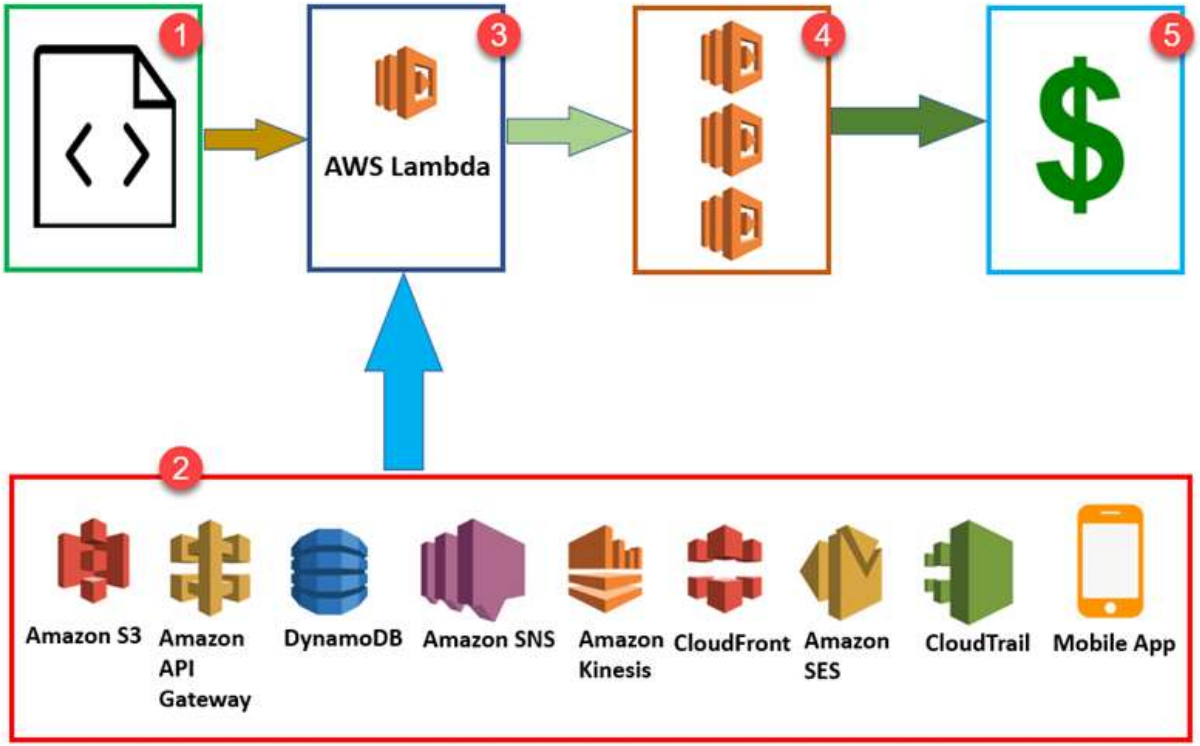
AWS Lambda

- **AWS Lambda** is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and SaaS applications.
- AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.
- You pay only for the compute time you consume - there is no charge when your code is not running

AWS Lambda

- To get working with **AWS Lambda**, we just have to push the code in AWS Lambda service. All other tasks and resources such as infrastructure, operating system, maintenance of server, code monitoring, logs and security is taken care by AWS.
- The code is executed based in response to events in AWS services such as adding/removing files in S3 bucket, updating Amazon DynamoDB tables, sending SNS notifications etc.
- AWS Lambda supports languages such as Java, Node.js, Python, C#, Go, Ruby

How Lambda works ?



When to use AWS Lambda ?

You can use AWS Lambda to create new backend application services triggered on demand using the Lambda application programming interface (API) or custom API endpoints built using Amazon API Gateway.

Some popular use-cases for using Lambda are:

1. Serverless Website
2. Document Conversion at Faster Pace
3. Log Analysis
4. Real-Time Data Processing
5. Predictive Page Rendering
6. Mass Emailing
7. Efficient Monitoring
8. Building Serverless Chatbots

AWS Lambda triggers

- Entry into an S3 object
- Inserting, updating and deleting data in Dynamo DB table
- Push notifications from SNS
- GET/POST calls to API Gateway
- Log entries in AWS Kinesis data stream
- Log history in CloudTrail

and many more..

AWS Lambda function configurations

- **General Configuration**
  - Memory: default 128 MB (CPU is proportional to memory)
  - Ephemeral storage: default 512 MB (/tmp space allocated for the function)
  - Timeout: default 3 sec

- **Triggers** – you can add triggers here. Triggers are events that fire the lambda function.

- **Permissions** – you can manage permissions such as execution role and policies attached to the function.

- **Environment Variables** – you can use these to pass run-time values to the function.
  - Python: `region = os.environ['AWS_REGION']`

- **Destinations** – to add destinations to your function to retain records of your function invocations or to retain discarded events. You can have SNS, SQL, EventBridge and Lambda as destinations.

- **VPC** – if you want to have your function access resources in a custom VPC

AWS Lambda Labs

Lab 1 : Lambda basics demo

- Instructions Document: L01-Lambda-Basics.txt
- Script File: cts-lambda-basics.py

Lab 2 : Lambda to log SNS messages to CloudWatch

- Instructions Document: L02-Lambda-SNS-Cloudwatch.txt
- Script File: cts-lambda-sns-cloudwatch.py

Lab 3 : Lambda to move files from one S3 bucket to another

- Instructions Document: L03-Lambda-S3-S3-Move.txt
- Script File: cts-lambda-s3-s3-move.py

Lab 4 : Lambda to delete files from S3 bucket

- Instructions Document: L04-Lambda-S3-Delete.txt
- Script File: cts-lambda-s3-delete.py

Amazon DynamoDB

SQL vs. NoSQL Databases

What is SQL?

SQL is a domain-specific language used to query and manage data. It works by allowing users to query, insert, delete, and update records in relational databases. SQL also allows for complex logic to be applied through the use of transactions and embedded procedures such as stored functions or views.

What is NoSQL?

NoSQL is a type of database that uses non-relational data structures, such as documents, graph databases, and key-value stores to store and retrieve data. NoSQL systems are designed to be more flexible than traditional relational databases and can scale up or down easily to accommodate changes in usage or load. This makes them ideal for use in applications

SQL vs. NoSQL Databases

SQL	NoSQL
Relational database management system (RDBMS)	Non-relational database management system
Suitable for structured data with predefined schema	Suitable for unstructured and semi-structured data
Data is stored in tables with columns and rows	Data is stored in collections or documents
Follows ACID properties for transaction management	Does not necessarily follow ACID properties
Supports JOIN and complex queries	Does not support JOIN and complex queries
Uses normalized data structure	Uses denormalized data structure
Requires vertical scaling to handle large volumes of data	Horizontal scaling is possible to handle large volumes of data
Examples: MySQL, PostgreSQL, Oracle, SQL Server	Examples: MongoDB, Cassandra, Amazon DynamoDB

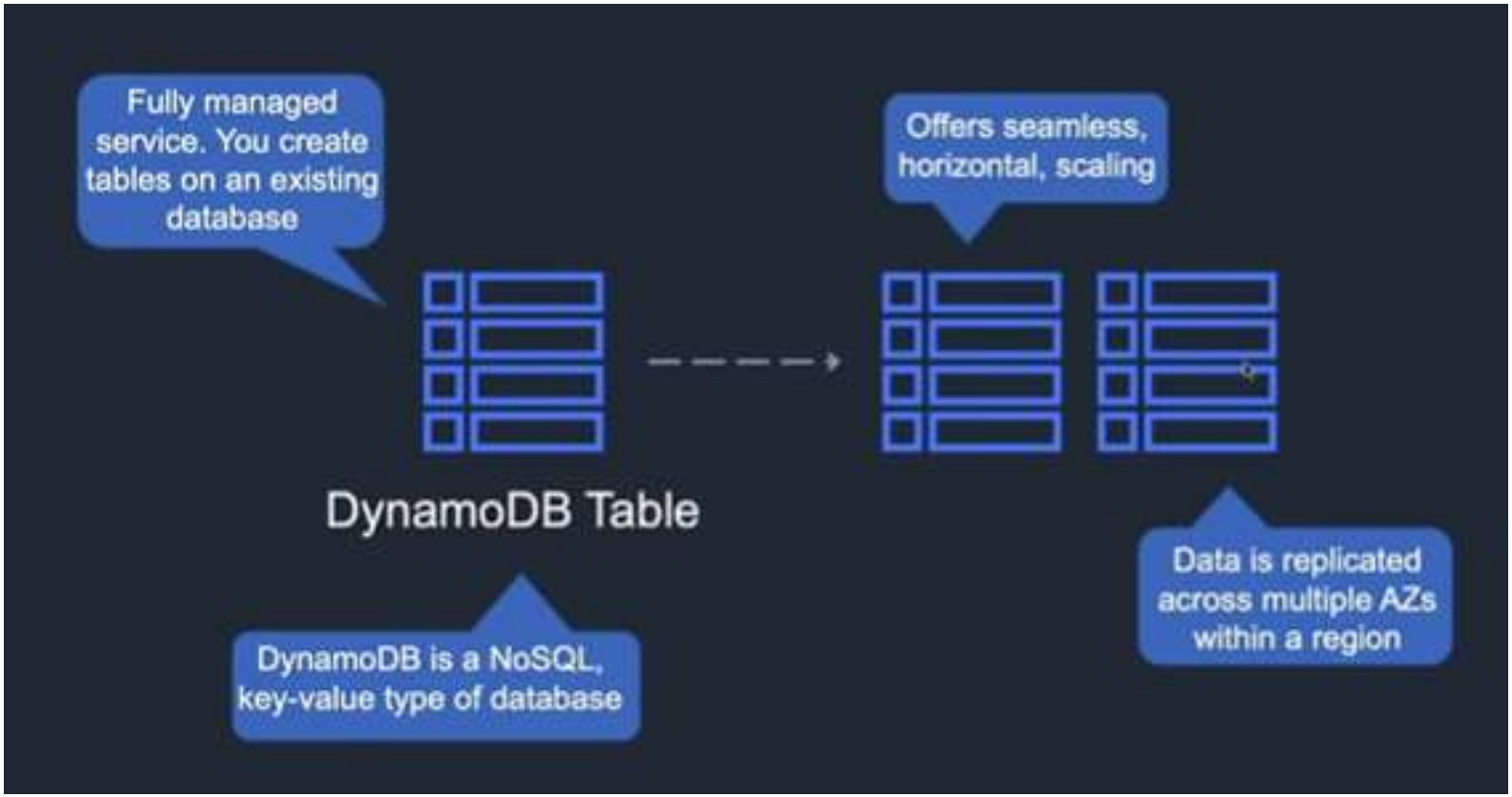
Amazon DynamoDB

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.
- DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

Amazon DynamoDB

- With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic.
- 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 utilization and performance metrics.
- DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Amazon DynamoDB



Amazon DynamoDB Components

- **Tables, Items & Attributes**
  - **Table**: A *table* is a collection of data.
  - **Item**: An *item* is a group of attributes that is uniquely identifiable among all of the other items. ***(each JSON inside the table)***
  - **Attributes**: Each item is composed of one or more attributes. An *attribute* is a fundamental data element, something that does not need to be broken down any further.
- **Primary Key**
  - Primary Key is mandatory for every table.
  - The primary key uniquely identifies each item in the table, so that no two items can have the same key. *(PersonId in the example)*

People

Amazon DynamoDB Components – Primary Key

- DynamoDB supports two different kinds of primary keys:
  - **Partition key**
    - A simple primary key, composed of one attribute known as the *partition key*.
    - Partition key's value is used as input to an internal hash function which determines the partition (physical storage internal to DynamoDB) in which the item will be stored.
    - In a table that has only a partition key, no two items can have the same partition key value.
  - **Partition key and sort key**
    - Is a *composite primary key with t*wo attributes - *partition key* and *sort key*.
    - In a table that has a partition key and a sort key, it's possible for multiple items to have the same partition key value. However, those items must have different sort key values.
- These keys can only be of **String, Binary** or **Number** data types.

Amazon DynamoDB Components – Indexes

- You can create one or more secondary indexes on a table. A *secondary index* lets you query the data in the table using an alternate key, in addition to queries against the primary key.
- DynamoDB supports two kinds of indexes:
  - **Global secondary index** – An index with a partition key and sort key that can be different from those on the table.
  - **Local secondary index** – An index that has the same partition key as the table, but a different sort key.
- Each table in DynamoDB has a quota of 20 global secondary indexes (default quota) and 5 local secondary indexes.

Working with AWS CLI for DynamoDB

- Make sure you installed AWS CLI
- Create an AWS user profile using Access keys: (specify the region)

```
kanak@kanak-VB:~$ aws configure --profile ctsdemouser
AWS Access Key ID [None]: [redacted]
AWS Secret Access Key [None]: [redacted]
Default region name [None]: us-east-1
Default output format [None]:
kanak@kanak-VB:~$
```

- Refer to Lab 2 to use CLI commands for creating a table, loading data into the table, scanning and querying the table and other operations.

Amazon DynamoDB Labs

Lab 1 - Amazon DynamoDB Basic Operations

Instructions Document: L01-DynamoDB-Basics.txt

In this lab, we explore how to use AWS DynamoDB management console to

- Create a DynamoDB table
- Add items to the DynamoDB table
- Scan the DynamoDB table
- Query the DynamoDB table.

Lab 2 – Using AWS CLI to work with Amazon DynamoDB

Instructions Document: L02-DynamoDB-CLI.txt

In this lab, we use AWS CLI to work with DynamoDB.

We use AWS CLI from a local terminal window to:

- Create a DynamoDB table
- Add items to the DynamoDB table from a local JSON file
- Scan the DynamoDB table

Lab 3 – Creating DynamoDB table using Lambda

Instructions Document: L03-DynamoDB-CreateTable-Lambda.txt

In this lab, we create a lambda function using boto3 to create a DynamoDB table.

Lab 4 – Put item into DynamoDB table using Lambda

Instructions Document: L04-DynamoDB-PutItem-Lambda.txt

In this lab, we create a lambda function using boto3 to insert an item into a DynamoDB table.

Lab 5 – Load data from S3 to DynamoDB using Lambda

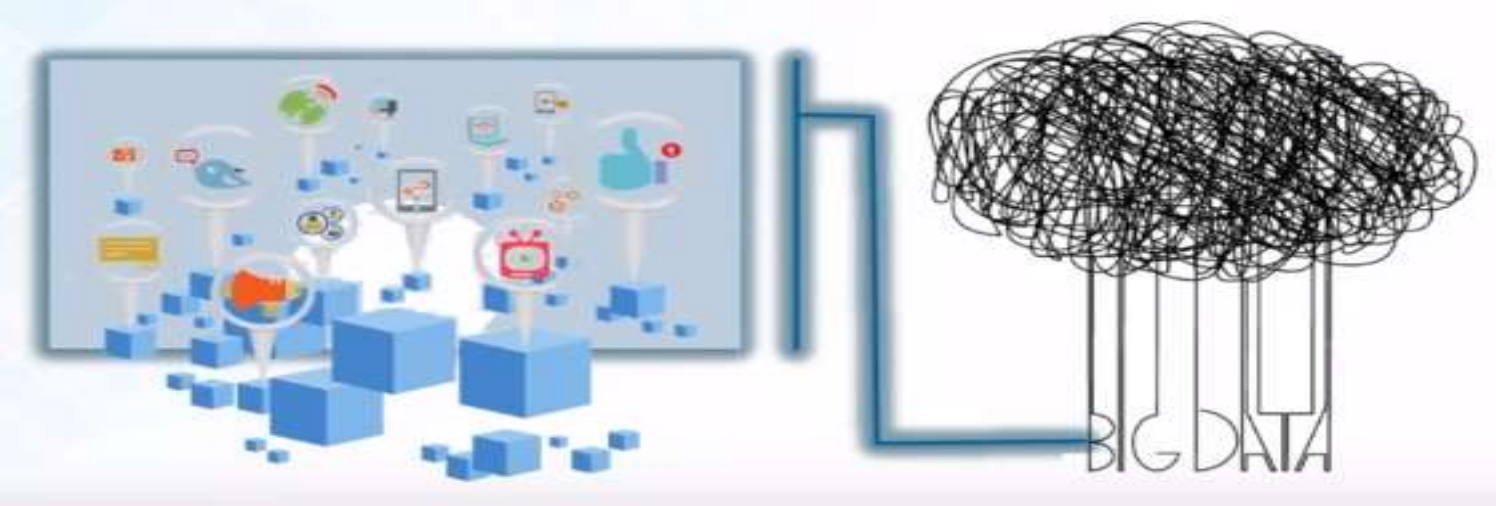
Instructions Document: L05-Load-data-from-S3-to-DynamoDB.txt

In this lab, we create a lambda function using boto3 which is triggered from the files uploaded to an S3 bucket. As we upload CSV files to the S3 bucket, the data is read by the lambda function and items are inserted into the DynamoDB table.

Amazon Elastic MapReduce (EMR)

What is Big Data ?

Big Data is a term for a collection of data sets so **large and complex**, that it becomes **difficult to store and process** using on-hand database management tools or traditional data processing applications



Challenges in Big Data

1. Storage

- Storing exponentially growing huge datasets using traditional ways is a big problem.

2. Processing complex data

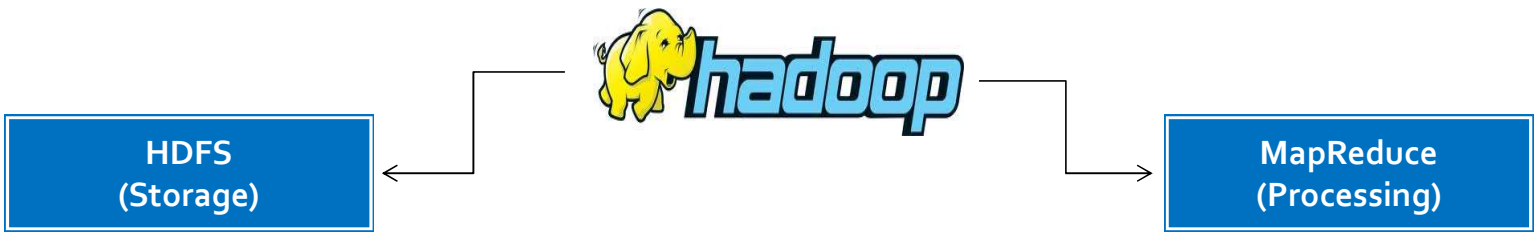
- Processing the data that has complex structure is a problem as the data comes as unstructured and semi-structured forms.

3. Processing data faster

- The data is growing at much faster rate than disk IO speeds. Bringing huge amount of data to the computation unit becomes a bottleneck.

What is Hadoop?

Hadoop is an open-source software framework for storage and processing of datasets on clusters of commodity hardware.



Allows to store any kind of data in a cluster in a distributed fashion

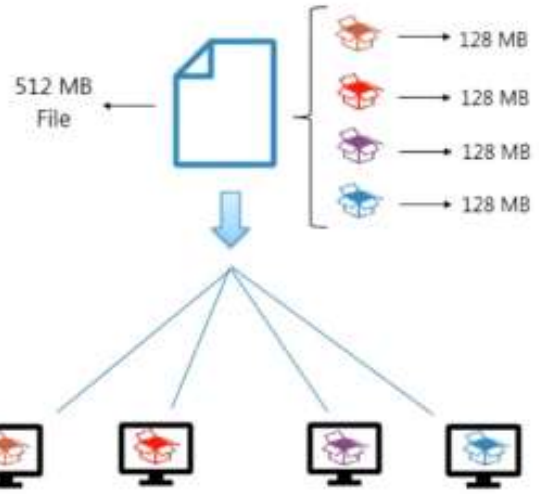
Allows parallel processing of data stored in HDFS

The Hadoop Solution

Problem 1: Storing exponentially growing huge datasets

Solution: HDFS

- Hadoop's storage unit
- Divides files into blocks
- Stores blocks across the cluster
- Replicates blocks as a fail-safe mechanism
- Horizontally Scalable



The Hadoop Solution

Problem 2: Storing unstructured data

Solution: HDFS

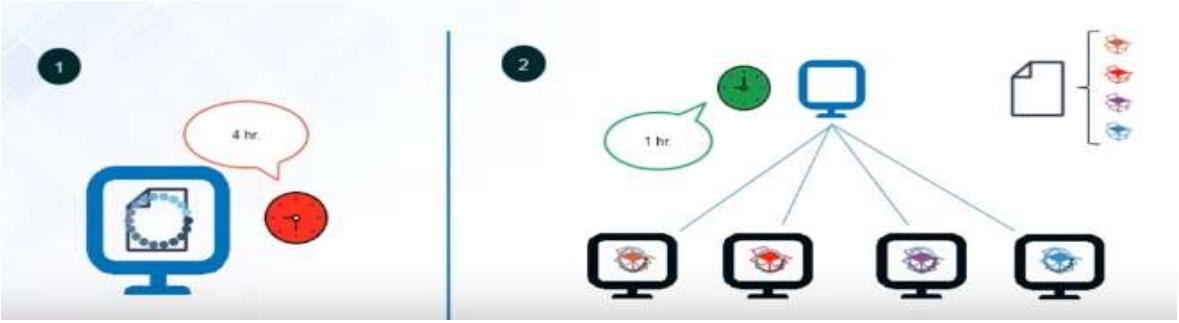
- HDFS allows to store any kind of data – be it structured or not.
- No schema validation done while writing data
- Follows write once and read many times paradigm (WORM)

The Hadoop Solution

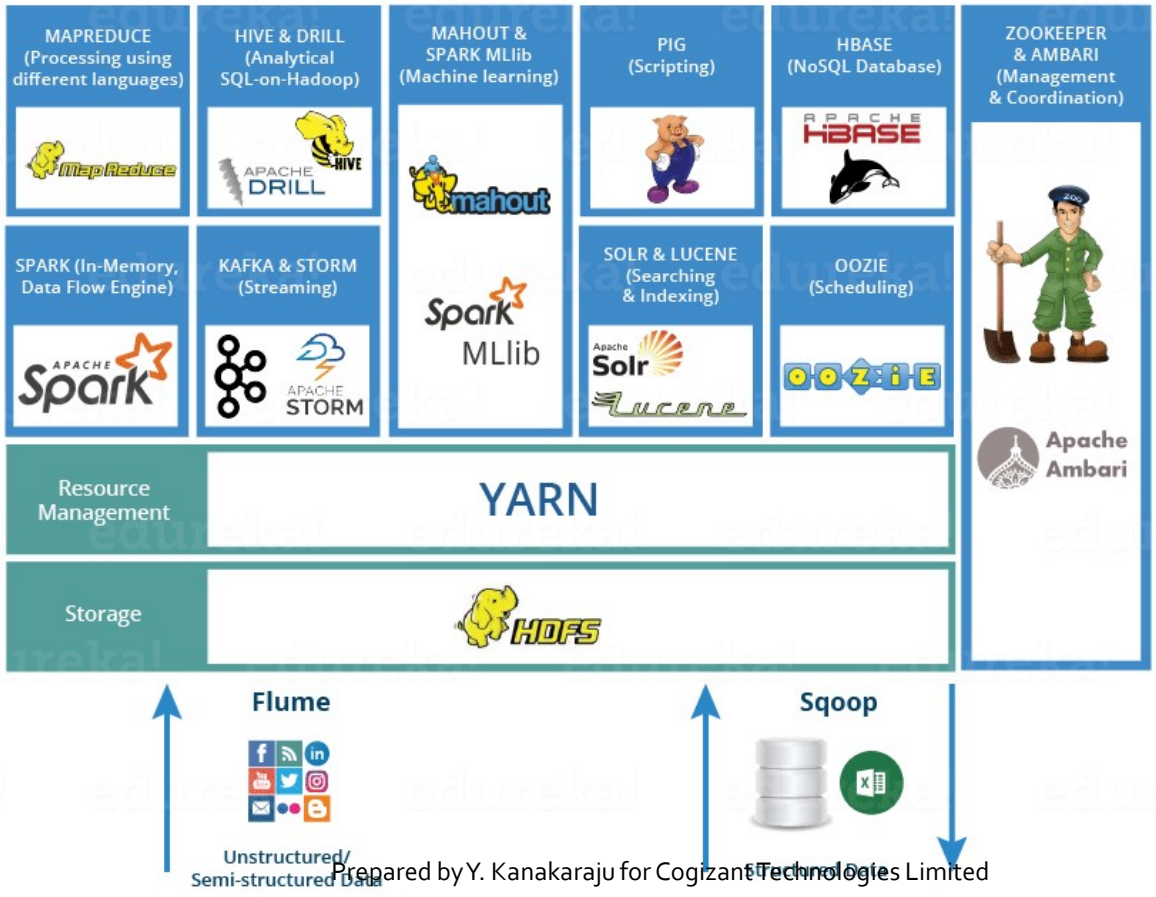
Problem 3: Processing data faster

Solution: MapReduce

- Provides parallel processing of data present in HDFS
- Allows data to be processed locally on each node making use of its resources.
- Brings processing to the data



Hadoop Ecosystem



Amazon EMR

Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as **Apache Hadoop** and **Apache Spark**, on AWS to process and analyze vast amounts of data

- Amazon EMR is a cloud big data platform for running large-scale distributed data processing jobs, interactive SQL queries, and machine learning applications using open-source analytics frameworks such as Apache Spark, Apache Hive, and Presto.

- EMR Uses:
  - S3 for storage
  - EC2 instances for computing

Benefits of Amazon EMR

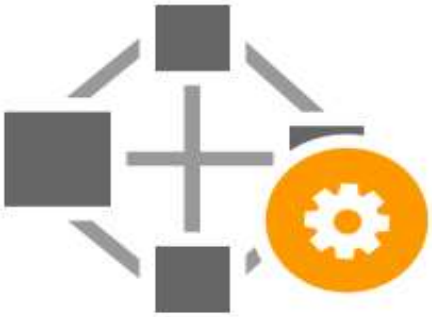
- **Reduces the cost of physical infrastructure** as there are no upfront costs involved related to purchasing hardware, software, maintenance and administration. You only pay per use.
- **Resource Utilization** – in EMR, storage and compute can be decoupled. You can spin up EC2 instances and cluster when needed without having to worry about data. You can achieve optimum resource utilization as you do not have to keep your cluster idle.
- **Saves Time** related to system management and administration.

How Amazon EMR works?

Upload

Upload your data and processing application to S3.

Create

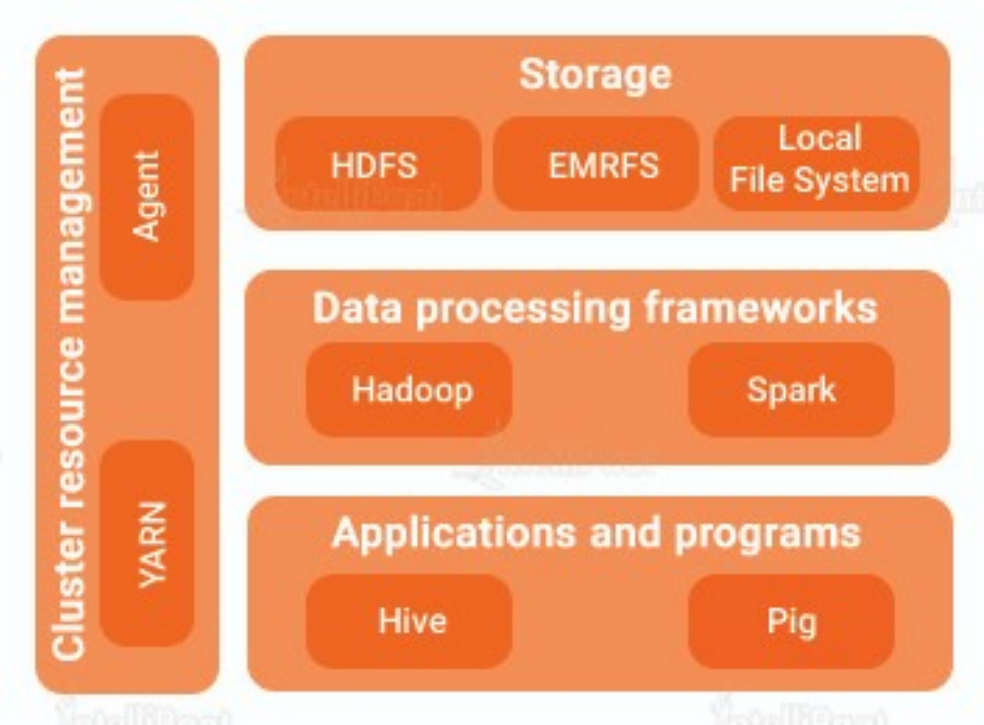


Configure and create your cluster by specifying data inputs, outputs, cluster size, security settings, etc.

Monitor

Monitor the health and progress of your cluster. Retrieve the output in S3.

EMR Architecture – Components of EMR



EMR Architecture – cluster & nodes

- The central component of Amazon EMR is the **cluster**.
  - A cluster is a collection Amazon EC2 instances.

- Each instance in the cluster is called a **node**.
  - Each node has a role within the cluster, referred to as the node type.
  - Amazon EMR also installs different software components on each node type, giving each node a role in a distributed application like Apache Hadoop.

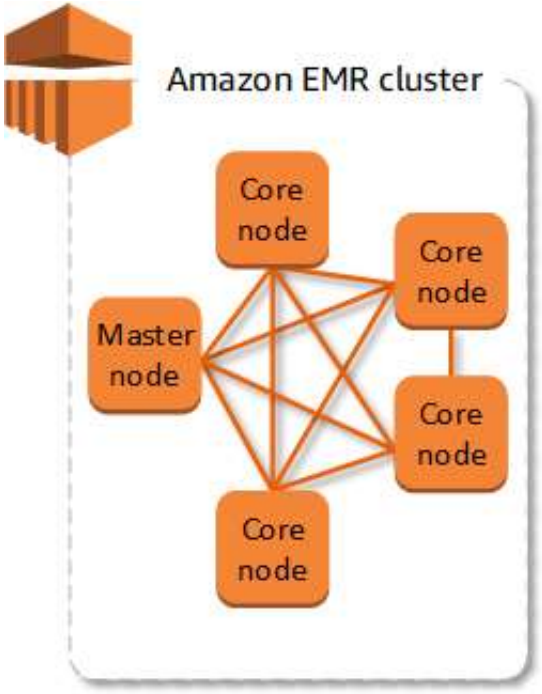
EMR Architecture – Types of nodes

The node types in Amazon EMR are as follows:

- **Primary node**: A node that manages the cluster by running software components to coordinate the distribution of data and tasks among other nodes for processing.
  - The primary node tracks the status of tasks and monitors the health of the cluster.
  - Every cluster has a primary node. We can create a single-node cluster with only the primary node.

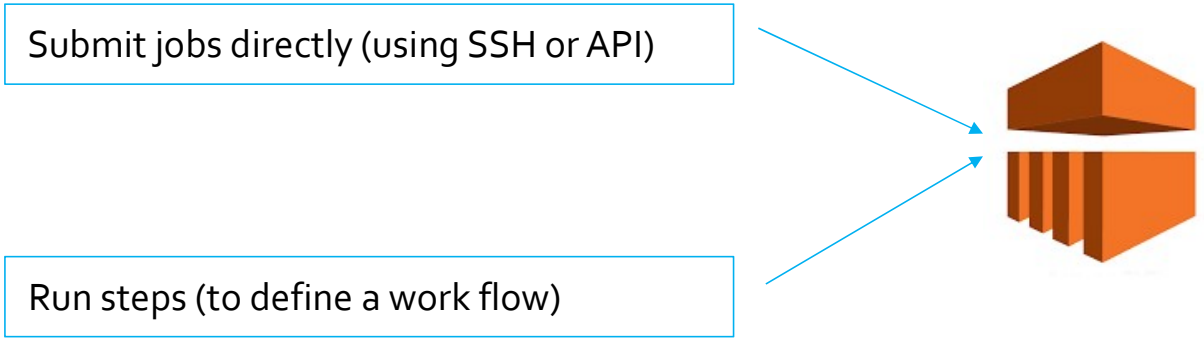
- **Core node**: A node with software components that run tasks and store data in HDFS on your cluster. Multi-node clusters have at least one core node.

- **Task node**: A node with software components that only runs tasks and does not store data in HDFS. Task nodes are optional.



Processing data on EMR cluster

When you launch your cluster, you choose the frameworks and applications to install for your data processing needs. To process data in your Amazon EMR cluster, you can submit jobs or queries directly to installed applications, or you can run *steps* in the cluster.



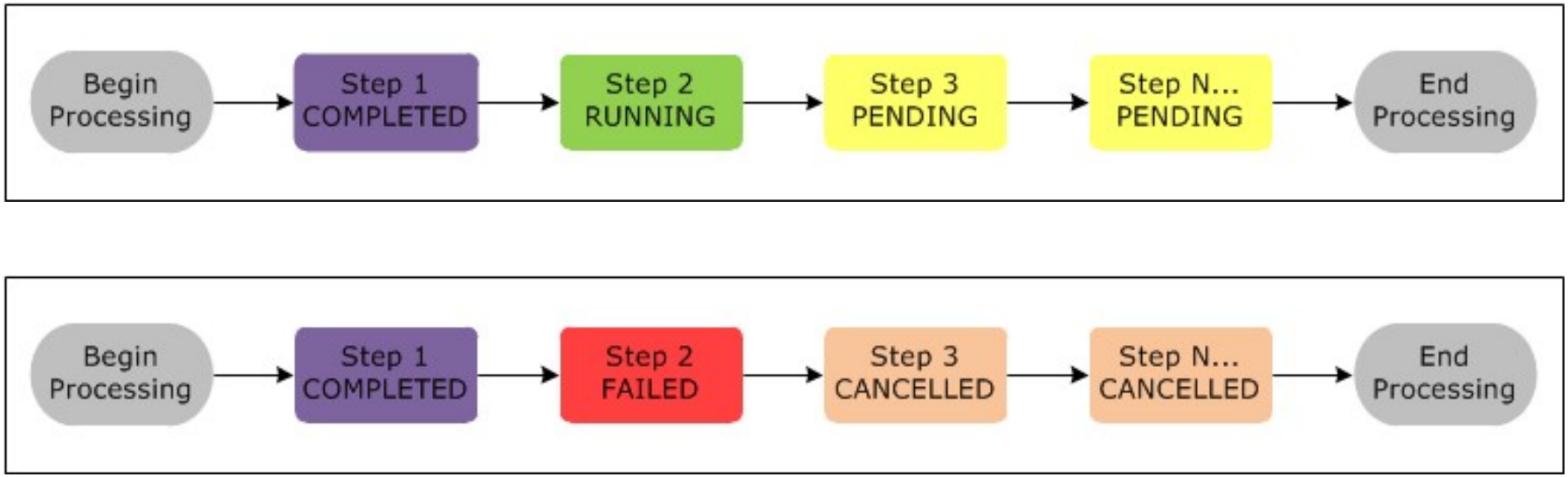
Submitting jobs directly to applications

You can submit jobs and interact directly with the software that is installed in your Amazon EMR cluster.

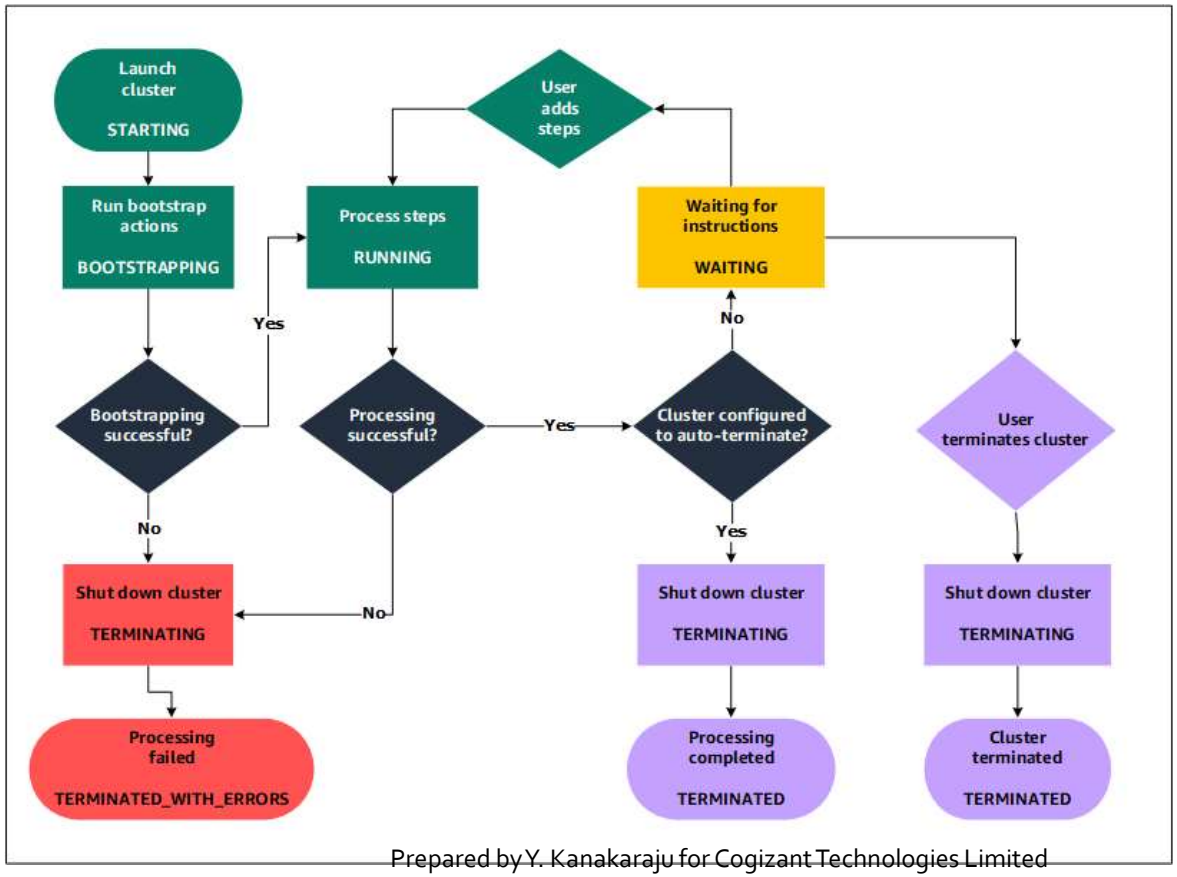
To do this, you typically connect to the primary node over a secure connection (SSH) and access the interfaces and tools that are available for the software that runs directly on your cluster.

Running steps to process data

You can submit one or more ordered steps to an Amazon EMR cluster. Each step is a unit of work that contains instructions to manipulate data for processing by software installed on the cluster.



Amazon EMR cluster life cycle



Amazon EMR Labs

Lab 1 - Amazon EMR basic operations

Instructions Document: L01-EMR-Basics.txt

In this lab, we perform the following operations:

- Spin up an EMR cluster
- Connect to the master node using SSH
- Test EMR cluster by running some HDFS and Hive operations
- Copy a file from S3 to EMRFS by adding a step (**`s3-dist-cp`**)

Lab 2 – Run a hive script using step execution on EMR cluster

Instructions Document: L02-Run-Hive-Script-Using-Step-Execution.txt

In this lab, we perform the following operations:

- Spin up an EMR cluster
- Upload Hive script and input files to an S3 bucket
- Run the hive script using a step in EMR and store the output in an S3 bucket

Lab 3 – Hive & DynamoDB integration on EMR

Instructions Document: L03-Hive-DynamoDB.txt

In this lab, we perform the following operations:

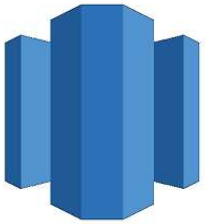
- Spin up an EMR cluster and SSH to master node
- Import data from S3 into EMR master node
- Create an DynamoDB Table
- Create a Hive external table using DynamoDBStorageHandler
- Load the data into DynamoDB table use Hive external table
- Query DynamoDB table from Hive.

Lab 4 – Running PySpark on EMR

Instructions Document: L04-Running-PySpark-on-EMR.txt

In this lab, we perform the following operations:

- Spin up an EMR cluster (with Spark) and SSH to master node
- Upload the PySpark scripts to S3
- Run PySpark script from EMR master node console using `spark-submit` command



Amazon Redshift

Understanding Data Warehouse

- A Data Warehouse stores data from multiple sources for analytical purposes.
- Data Warehouses are designed for OLAP which is used to integrate copies of transactional data from other systems and use it for analytical purposes.

Data warehouse technology options

There are three main options available for building a data warehouse:

- Row-oriented databases
- Column-oriented databases
- Massively parallel processing (MPP) architectures.

Data warehousing: Row-oriented databases

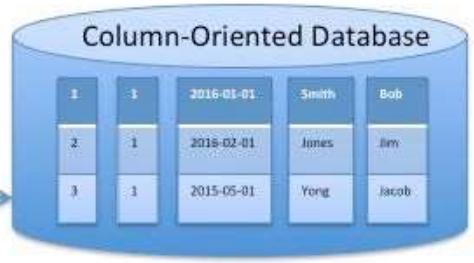
- Row-oriented databases typically store whole rows in a physical block. High performance for read operations is achieved through secondary indexes.
- Databases such as Oracle Database Server, Microsoft SQL Server, MySQL, and PostgreSQL are row-oriented database systems. These systems have been traditionally used for data warehousing, but they are better suited for transactional processing (OLTP) than for analytics.
- In a row-based data warehouse, every query has to read through all of the columns for all of the rows in the blocks that satisfy the query predicate, including columns you didn't choose. This approach creates a significant performance bottleneck in data warehouses, where your tables have more columns, but your queries use only a few.

Data warehousing: Column-oriented databases

- Column-oriented databases organize each column in its own set of physical blocks instead of packing the whole rows into a block. This functionality allows them to be more I/O efficient for read-only queries, because they have to read only those columns accessed by a query from disk (or from memory).
- This approach makes column-oriented databases a better choice for data warehousing.
- After faster I/O, the next biggest benefit to using a column-oriented database is improved compression. Because every column is packed into its own set of blocks, every physical block contains the same data type. When all the data is the same data type, the database can use extremely efficient compression algorithms.
- Some column-oriented databases that are used for data warehousing include Amazon Redshift, Vertica, Greenplum, Teradata Aster, Netezza, and Druid.

Row-oriented vs. Column-oriented databases

EmpNo	DeptId	Hire_Date	LastName	FirstName
1	1	2016-01-01	Smith	Bob
2	1	2016-02-01	Jones	Jim
3	1	2015-05-01	Yong	Jacob
4	2	2013-06-28	David	Stein
5	2	2016-01-17	Scott	Jack
6	3	2003-09-11	Brian	Davis



Data warehousing: MPP architectures

- An MPP architecture enables you to use all the resources available in the cluster for processing data, which dramatically increases performance of petabyte scale data warehouses.
- MPP data warehouses allow you improve performance by simply adding more nodes to the cluster.
- Amazon Redshift, Druid, Vertica, Greenplum, and Teradata Aster are some of the data warehouses built on an MPP architecture.
- Open-source frameworks such as Hadoop and Spark also support MPP.

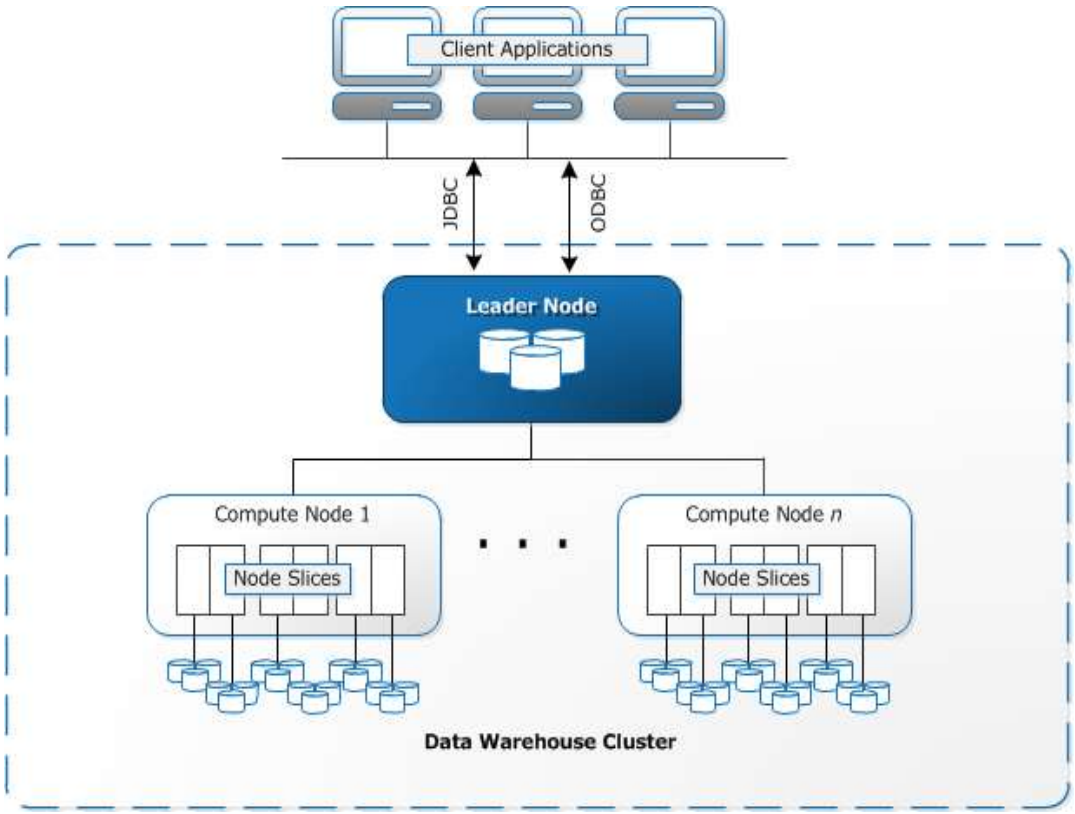
Amazon Redshift

- Amazon Redshift is a fully managed, petabyte-scale **data warehouse** service in the cloud. Amazon RedShift uses **columnar database with MPP architecture**
- Redshift data warehouse is a collection of computing resources called **nodes,** which consists of a **leader** node and **one or more compute nodes,** which are organized into a group called a **cluster**.
  - The type and number of compute nodes that you need depends on the size of your data, the number of queries you will execute, and the query execution performance that you need.
  - Each cluster runs an Amazon Redshift engine and contains one or more databases.
- Amazon Redshift integrates with various data loading and ETL tools and BI reporting, data mining, and analytics tools.
- Amazon Redshift is **based on industry-standard PostgreSQL**.

Amazon Redshift cluster

- The core infrastructure component of an Amazon Redshift data warehouse is a cluster. A cluster is composed of one or more compute nodes.
- If a cluster is provisioned with two or more compute nodes, an additional leader node coordinates the compute nodes and handles external communication.
- Your client application interacts directly only with the leader node. The compute nodes are transparent to external applications.

Amazon Redshift architecture



Amazon Redshift architecture – Leader & Compute nodes

- **Leader Node**
  - The leader node manages communications with client programs and all communication with compute nodes.
  - It parses and develops execution plans to carry out database operations, in particular, the series of steps necessary to obtain results for complex queries. Based on the execution plan, the leader node compiles code, distributes the compiled code to the compute nodes, and assigns a portion of the data to each compute node.
  - The leader node distributes SQL statements to the compute nodes only when a query references tables that are stored on the compute nodes. All other queries run exclusively on the leader node.

Amazon Redshift architecture – Leader & Compute nodes

- **Compute Node**
  - The leader node compiles code for individual elements of the execution plan and assigns the code to individual compute nodes. The compute nodes execute the compiled code and send intermediate results back to the leader node for final aggregation.
  - Each compute node has its own dedicated CPU, memory, and attached disk storage, which are determined by the node type. As your workload grows, you can increase the compute capacity and storage capacity of a cluster by increasing the number of nodes, upgrading the node type, or both.
  - Redshift provides several node types for your compute and storage needs – RA3, DC2 (and DS2 which generally is not recommended)

Redshift SORTKEY

- Because Redshift is a columnar database with compressed storage, it doesn't use indexes like transactional databases such as MySQL and Oracle. Instead, it uses DISTKEYs and SORTKEYs.
- Redshift **SORTKEY** determines the order in which rows in a table are stored.
- Query performance is improved when SORTKEYs are properly used as it enables the query optimizer to read fewer chunks of data filtering out the majority of it.
- There can be multiple columns defined as SORTKEYs. Data stored in the table can be sorted using these columns. The query optimizer uses this sort ordered table while determining optimal query plans.

Redshift SORTKEY

Amazon Redshift supports two kinds of Sort Keys:

- **Compound Sort Keys**
  - These are made up of all the columns that are listed in the Redshift sort keys definition during the creation of the table, in the order that they are listed. Therefore, it is advisable to put the most frequently used column at the first in the list.

  - COMPOUND is the default sort type. Compound sort keys might speed up JOINs, GROUP BY and ORDER BY operations, and window functions that use PARTITION BY.

- **Interleaved Sort Keys**
  - Interleaved sort gives equal weight to each column in the Redshifts sort keys. As a result, it can significantly improve query performance where the query uses restrictive predicates (equality operator in WHERE clause) on secondary sort columns.

Redshift DISTKEY

Redshift **DISTKEY** (**Distribution Keys)** determine where data is stored in Redshift. Clusters store data fundamentally across the compute nodes. Query performance suffers when a large amount of data is stored on a single node.

Uneven distribution of data across computing nodes leads to the skewness of the work a node has to do and you don't want an under-utilized compute node. So the distribution of the data should be uniform. Distribution is per table. So you can select a different distribution style for each of the tables you are going to have in your database.

Redshift distribution styles

When you create a table, you can designate one of four distribution styles; AUTO, EVEN, KEY, or ALL

- **AUTO**
  With AUTO distribution, Amazon Redshift assigns an optimal distribution style based on the size of the table data. The change in distribution style occurs in the background with minimal impact to user queries.

- **EVEN**
  The leader node distributes the rows across the slices in a round-robin fashion, regardless of the values in any particular column. EVEN distribution is appropriate when a table doesn't participate in joins. It's also appropriate when there isn't a clear choice between KEY distribution and ALL distribution.

- **KEY**
  The rows are distributed according to the values in one column. The leader node places matching values on the same node slice. If you distribute a pair of tables on the joining keys, the leader node collocates the rows on the slices according to the values in the joining columns.

- **ALL**
  A copy of the entire table is distributed to every node. ALL distribution ensures that every row is collocated for every join that the table participates in.

SORTKEY & DISTKEY examples

```
create table errors (
  err_code int,
  created_at timestamp
) distkey(err_code) sortkey(err_code);
```

```
create table t1(col1 int distkey, col2 int) diststyle key;
```

```
create table t2(col1 int distkey sortkey, col2 int);
```

```
create table t3(col1 int, col2 int, col3 varchar(10))
diststyle all
interleaved sortkey (col1, col2);
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't2';
```

Best practices: Splitting data into multiple files

- While loading very large files into Redshift cluster, it is advised to split files into multiple splits. By having multiple splits we can leverage the parallel processing of files using multiple slices of the compute nodes.
    - You can split the files by having common prefix for the filenames or by explicitly listing the files in a manifest file.
- Divide your data file into splits in such a way that the number of splits is a multiple of number slices in your cluster.
    - For example, ds2.xl has two slices per compute node
    - If your cluster has two ds2.xl nodes, split your data into 4 files or multiple of 4.

Best practices: Selecting keys

- Pick a small number of important queries you want to optimize your databases for. You can't optimize your table for all queries, unfortunately.
- To avoid a large data transfer over the network, define a DISTKEY.
- From the columns used in your queries, choose a column that causes the least amount of skew as the DISTKEY. A column that has many distinct values, such as timestamp, would be a good first choice. Avoid columns with few distinct values, such as credit card types or days of the week.
- Even though it will almost never be the best performer, a table with no DISTKEY/SORTKEY is a decent all-around performer. It's a good option not to define DISTKEY and SORTKEY until you really understand the nature of your data and queries.

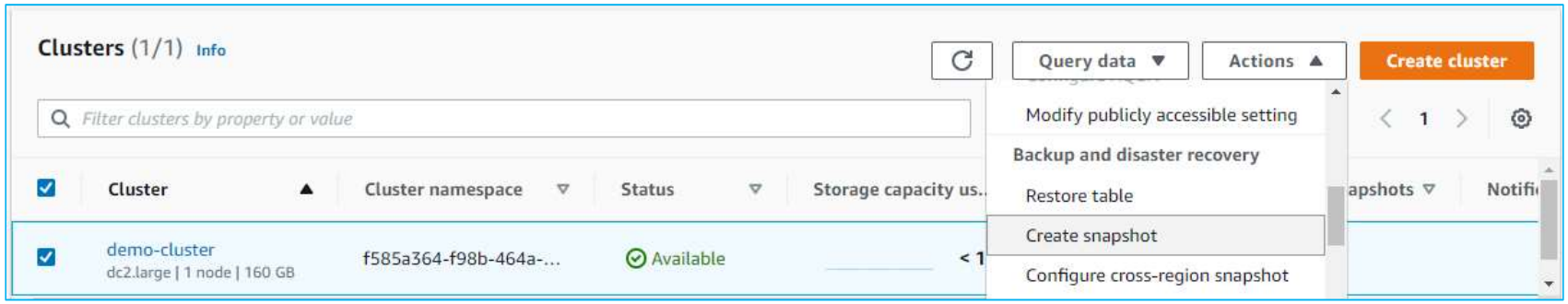
Creating a snapshot of a Redshift cluster

- Snapshots are point-in-time backups of a cluster. There are two types of snapshots: ***automated*** and ***manual***.

- Amazon Redshift stores these snapshots internally in Amazon S3 by using an encrypted SSL connection.

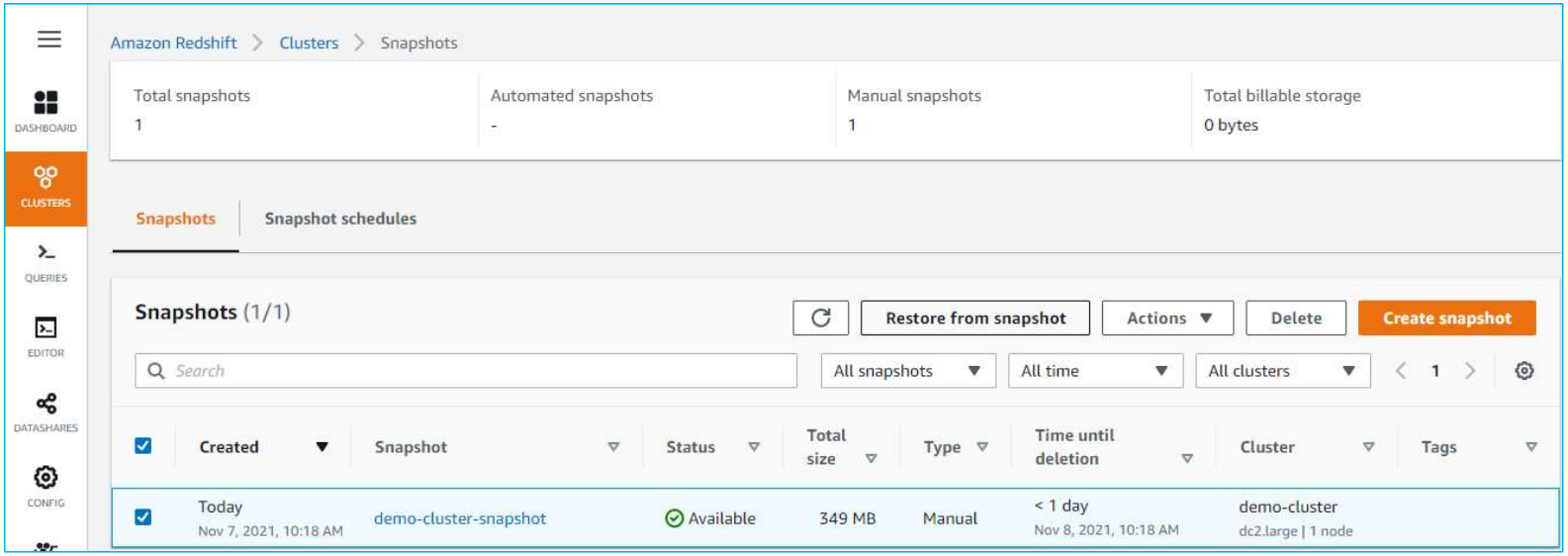
Creating a snapshot of a Redshift cluster

- You can create a snapshot of a cluster either explicit or while deleting the cluster.
- To create a snapshot, select the cluster, go to Actions menu and select create snapshot option. Mention the retention period and other details and create.
- Other way to create a snapshot is while deleting the cluster. Mention a snapshot name while deleting the cluster. The snapshot will be created and then the cluster is deleted.



Restoring a cluster from a snapshot

- Go to Redshift Dashboard. Select Clusters >> Snapshots option from the menu.
- Select the snapshot and click on **Restore from snapshot** button.



EMR vs. Redshift – When to use what?

- **Use Redshift when**
  - Traditional data warehouse
  - When you need the data relatively hot for analytics such as BI
  - When there is no data engineering team
  - When you require joins
  - When u need a cluster 24X7

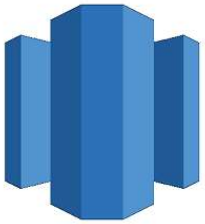
- **Use EMR (Spark, Presto, Hive) when**
  - When you don't need a cluster 24X7
  - When elasticity is important (auto scaling on tasks)
  - When cost is important: spots
  - Until a few hundred TB's, in some cases PB's will work.
  - When you want to separate compute and storage (external table + task node + auto scaling)

Public S3 buckets

- There are a few public S3 buckets that we can use to practice loading data to Redshift.
- Query the following S3 buckets using AWS CLI commands.

```
aws s3 ls s3://awssampledbuswest2/tickit/

aws s3 ls s3://awssampledbuswest2/ssbgz/
```



Amazon Redshift Labs

Lab 1 – Create a Redshift cluster and perform basic operations

Instructions Document: L01-Redshift-Basics.txt

In this lab, we perform the following operations:

- Create a one node Redshift cluster with sample data
- Run queries using 'Query Editor' on existing data
- Create a new database, schema and a table.
- Load data into the Redshift table from S3 (or from a local file)

Lab 2 – Working with Redshift COPY command

Instructions Document: L02-Redshift-Copy-Command.txt

In this lab, we perform the following operations:

- Work with COPY command using key-based authentication
- Work with COPY command using role-based authentication

Lab 3 – Connect to Redshift cluster using SQLWorkbench/J client

Instructions Document: L03-Redshift-SQLWorkbench.txt

In this lab, we perform the following operations:

- Download and install SQLWorkbench/J
- Setup Amazon Redshift JDBC driver in SQLWorkbench/J
- Connect to the Redshift cluster from SQLWorkbench/J
- Perform database operations from SQLWorkbench/J using JDBC connection

Amazon Athena

Amazon Athena

- Amazon Athena is a serverless interactive query service or interactive data analysis tool which is used for processing complex queries and in a lesser amount of time. Due to its serverless feature, it needs no infrastructure to manage or to setup.
- It can quickly analyze the data with the help of Amazon S3 using standard SQL. It even does not need to load the data in Athena.
- All we require to do is to point to the data in Amazon S3, define the particular schema and start querying using the standard SQL. With the help of Amazon Athena, we can process any of data, whether it is structured, semi-structured or unstructured data, i.e., it can handle the data in CSV, Avro, Parquet and ORC.
- Athena uses S3 as its storage layer.

Amazon Athena

When to use Amazon Athena ?

- You should use Amazon Athena if you want to run interactive ad hoc SQL queries against data on Amazon S3, without having to manage any infrastructure or clusters.
    - Athena is useful if you want to run a quick query on web logs to troubleshoot a performance issue on your site. You just define a table for your data and start querying using standard SQL. You can analyze data stored in CSV, JSON, AVRO, Parquet and ORC data formats.
- Athena integrates with QuickSight for easy data visualization. You can use Athena to generate reports or to explore data with BI tools or SQL clients connected with a JDBC or ODBC driver.
- Athena integrates with the AWS Glue Data Catalog, which offers a persistent metadata store for your data in Amazon S3. This allows you to create tables and query data in Athena based on a central metadata store available throughout your AWS account and integrated with the ETL and data discovery features of AWS Glue.

Amazon Athena Labs

Lab 1 – Athena basic operations

Instructions Document: L01-Athena-Basics.txt

In this lab, we perform the following operations:

- Setup Output S3 bucket for Amazon Athena
- Create a Catalog table and database using Athena query editor using S3 as data source
- Run queries on the table using ANSI SQL
- Managing query and its results
- Download the results of the query in CSV format

Lab 2 – Create table using Glue catalog and Glue crawler

Instructions Document: L02-Athena-Glue-Catalog.txt

In this lab, we perform the following operations:

- Create a metadata table using Glue data catalog from S3 bucket
- Query the table from Athena
- Create a metadata table using Glue crawler by crawling an S3 bucket
- Query the table from Athena using ANSI SQL

Lab 3 – Create table using Hive DDL command

Instructions Document: L03-Athena-Hive-DDL.txt

In this lab, we perform the following operations:

- Create a metadata table in Athena query editor using Hive DDL command
- Query the data using Athena

AWS Glue

AWS Glue

- AWS Glue is a **serverless data integration and managed ETL service**.
- AWS Glue makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development.
- AWS Glue provides all the capabilities needed for data integration so that you can start analyzing your data and putting it to use in minutes instead of months.

Benefits of AWS Glue

- AWS Glue can scan through all the available data with a crawler and discovers and stores your schemas.
- Final processed data can be stored in many different places (Amazon RDS, Amazon Redshift, Amazon S3 etc.)
- It's a cloud service, hence no upfront investments required for on-premises infrastructure.
- It's a serverless ETL service, hence no provisioning of hardware required.
- It gives you the Python/Scala ETL code right off the bat.

When to use Glue?

- Build event-driven ETL pipelines

  AWS Glue can run your ETL jobs as new data arrives. For example, you can use an AWS Lambda function to trigger your ETL jobs to run as soon as new data becomes available in Amazon S3. You can also register this new dataset in the AWS Glue Data Catalog as part of your ETL jobs.

When to use Glue?

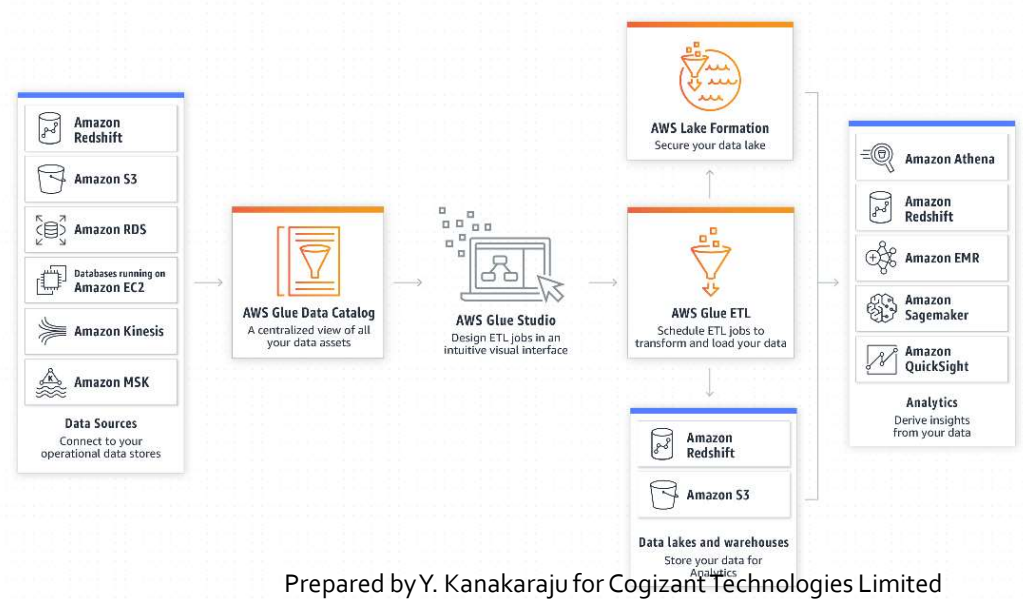
- Create a unified catalog to find data across multiple data stores

  You can use the AWS Glue Data Catalog to quickly discover and search across multiple AWS data sets without moving the data. Once the data is cataloged, it is immediately available for search and query using Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum.

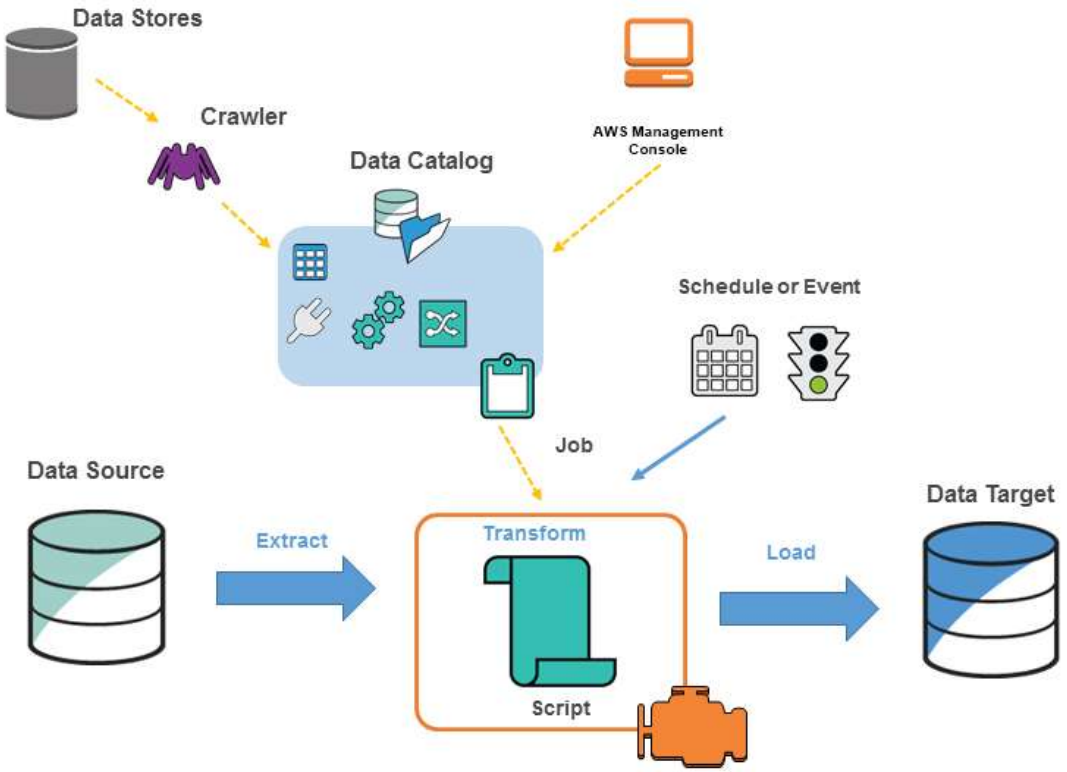
When to use Glue?

- Create, run, and monitor ETL jobs without coding

  AWS Glue Studio makes it easy to visually create, run, and monitor AWS Glue ETL jobs. You can compose ETL jobs that move and transform data using a drag-and-drop editor, and AWS Glue automatically generates the code. You can then use the AWS Glue Studio job run dashboard to monitor ETL execution.



AWS Glue Architecture



AWS Glue components

- **Data catalog**
  The data catalog holds the metadata and the structure of the data.
- **Database**
  It is used to create or access the database for the sources and targets.
- **Table**
  Create one or more tables in the database that can be used by the source and target.
- **Crawler**
  A crawler is used to retrieve data from the source using built-in or custom classifiers. It creates/uses metadata tables that are pre-defined in the data catalog.

AWS Glue components

- **Job**
  A job is business logic that carries out an ETL task. Internally, Apache Spark with python or scala language writes this business logic.

- **Trigger**
  A trigger starts the ETL job execution **on-demand** or on a **schedule** or when a **job or crawler event** is generated.

- **Development endpoint**
  Creates a development environment where the ETL job script can be tested, developed, and debugged.

- **Workflow**
  A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.

AWS Glue Labs

Lab 1 – Basics of a Glue Job (CSV to Parquet)

Instructions Document: L01-Glue-Basics-CSV-to-Parquet.txt

In this lab, we perform the following operations:

- Create Crawler and Catalog table for flights data
- Validate using Athena
- Create a Policy and a Role to access S3 from Glue
- Create a Glue Job to convert file format from CSV to Parquet
- Run and Monitor the job
- Create Catalog table on top of new location with parquet file format
- Validate new table using Athena

Lab 2 – Create and run a Glue trigger

Instructions Document: L02-Glue-Triggers.txt

In this lab, we perform the following operations:

- Create a Glue trigger to invoke a job
- Run the trigger to execute the job
- Validate the results using Athena

Lab 3 – Setup and run a Glue workflow

Instructions Document: L03-Glue-Workflow.txt

In this lab, we perform the following operations:

- Build the workflow for the pipeline
  - Crawl the source folder for flights data to refresh the table
  - Run Glue Job to convert the file format
  - Crawl the target folder to refresh the table
- Run and monitor the workflow
- Validate both source and target tables using Athena

Lab 4 – Glue ETL pipeline using PySpark - S3 to S3

Instructions Document: L04-Glue-S3-to-S3-ETL.txt

In this lab, we perform the following operations:

- Load the required data to S3
- Create an IAM Role with required policies
- Create and run a Glue crawler
- Create the Glue job and provide the ETL script (PySpark code)
- Run the job and validate the results

Lab 5 – Glue PySpark job with Joins

- Instructions Document: L04-Glue-ETL-PySpark-Joins.txt
- PySpark Script File: moviesratings-joins.py

In this lab, we perform the following operations:

- Load the required data to S3
- Create and run the Glue crawlers
- Create the Glue job and provide the PySpark script
- Run the job and validate the results

Amazon Kinesis

Streaming data analytics

Data that is generated continuously by thousands of data sources, which typically send in data records simultaneously, and in small sizes is called **Streaming data**.

- Streaming data include a wide variety of data such as:
    - log files generated by web sites and mobile apps
    - e-commerce purchases
    - in-game player activity
    - social network feeds
    - financial trading floors etc.

Amazon Kinesis

Amazon Kinesis is a fully managed service that makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information.

- Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application.
- With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications.
- Amazon Kinesis enables you to process and analyze data as it arrives and respond instantly instead of having to wait until all your data is collected before the processing can begin.

Amazon Kinesis Use-cases

- **Create real-time applications**
  Build apps for application monitoring, fraud detection, and live leaderboards. Analyze data and emit the results to any data store or application.

- **Evolve from batch to real-time analytics**
  Perform real-time analytics on data that has been traditionally analyzed using batch processing. Get the latest information without delay.

- **Analyze IoT device data**
  Process streaming data from IoT devices, and then use the data to programmatically send real-time alerts and respond when a sensor exceeds certain operating thresholds.

- **Build video analytics applications**
  Securely stream video from camera-equipped devices. Use streams for video playback, security monitoring, face detection, ML, and other analytics.

Amazon Kinesis Services

Kinesis Data Streams

Capture, process, and store data streams

Amazon Kinesis Data Streams is a scalable and durable real-time data streaming service that can continuously capture gigabytes of data per second from hundreds of thousands of sources.

Kinesis Data Firehose

Load data streams into AWS data stores

Amazon Kinesis Data Firehose is the easiest way to capture, transform, and load data streams into AWS data stores for near real-time analytics with existing business intelligence tools.

Kinesis Data Analytics

Analyze data streams with SQL or Apache Flink

Amazon Kinesis Data Analytics is the easiest way to process data streams in real time with SQL or Apache Flink without having to learn new programming languages or processing frameworks.

Kinesis Video Streams

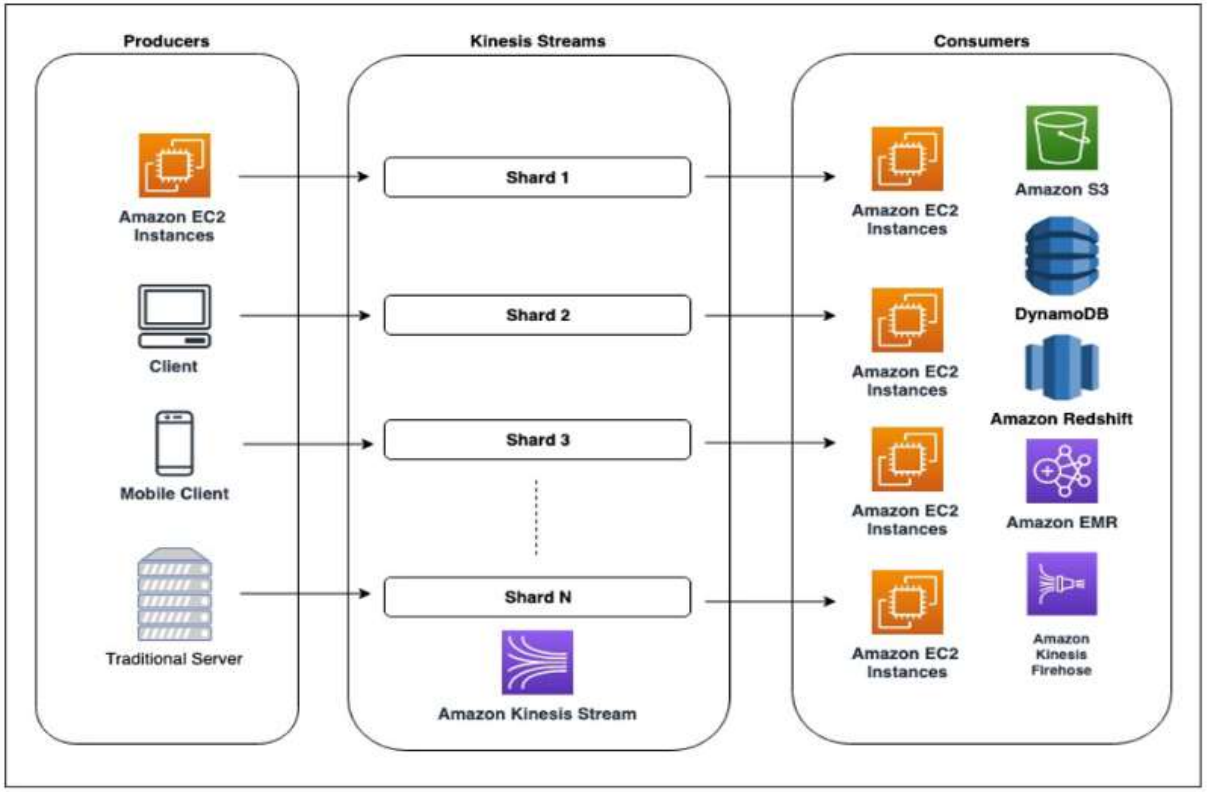
Capture, process, and store video streams

Amazon Kinesis Video Streams makes it easy to securely stream video from connected devices to AWS for analytics, machine learning (ML), and other processing.

Kinesis Data Streams

- Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service.
- KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website click streams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events.
- The data collected is available in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.

Kinesis Data Streams High-Level Architecture



Kinesis Data Streams Terminology

- **Data record**
  - The unit of data stored by Kinesis Data Streams is a data record.
- **Data stream**
  - A data stream represents a group of data records.
  - The data records in a data stream are distributed into shards.
- **Shards**
  - A shard has a sequence of data records in a stream.
  - When you create a stream, you specify the number of shards for the stream.
  - The total capacity of a stream is the sum of the capacities of its shards. You can increase or decrease the number of shards in a stream as needed. You are charged on a per-shard basis.

Kinesis Data Streams Terminology

- **Partition Key**
  - A partition-key is used to group data by shard within a stream
- **Sequence Number**
  - Each data record has a sequence number that is unique per partition-key within its shard.
  - Kinesis Data Streams assigns the sequence number after you write to the stream with `client.putRecords` or `client.putRecord`.
  - Sequence numbers for the same partition key generally increase over time.
  - The longer the time period between write requests, the larger the sequence numbers become.
- **Producers & Consumers**
  - A producer puts data records into shards and a consumer gets data records from shards.

Sending Data to Amazon Kinesis Data Streams

You can build producers for Kinesis Data Streams using :

- **AWS SDK Data Streams API**
  - Kinesis Data Streams provides two APIs for putting data into an Amazon Kinesis Stream: PutRecord and PutRecords
- **Kinesis Producer Library (KPL)**
  - KPL is a highly configurable library that helps you put data into an Kinesis data stream.
- **Kinesis Agent**
  - Kinesis Agent is a pre-built Java application that offers an easy way to collect and send data to your Amazon Kinesis stream. The agent monitors certain files and continuously sends data to your stream.

Processing Data from Amazon Kinesis Data Streams

You can easily process data with built-in integrations to AWS Lambda, Amazon Kinesis Data Analytics, Amazon Kinesis Data Firehose and KCL.

- **Amazon Kinesis Data Firehose**
  - Firehose is the easiest way to reliably transform and load streaming data into data stores and analytics tools. (such as S3, Redshift etc.)
  - You can use a Kinesis data stream as a source for a Kinesis Data Firehose.

- **Amazon Kinesis Data Analytics**
  - Kinesis Data Analytics is the easiest way to transform and analyze streaming data in real time with Apache Flink.
  - You can use a Kinesis data stream as a source and a destination for a Kinesis data analytics application.
  - You can use Amazon Kinesis Data Analytics Studio to interact with streaming data using SQL, Python and Scala.

Processing Data from Amazon Kinesis Data Streams

- **AWS Lambda**
  - You can subscribe Lambda functions to automatically read records off your Kinesis data stream.
  - AWS Lambda is typically used for record-by-record (also known as event-based) stream processing.

- **Amazon Kinesis Client Library (KCL)**
  - KCL is a pre-built library that helps you easily build Amazon Kinesis applications for reading and processing data from an Amazon Kinesis data stream.

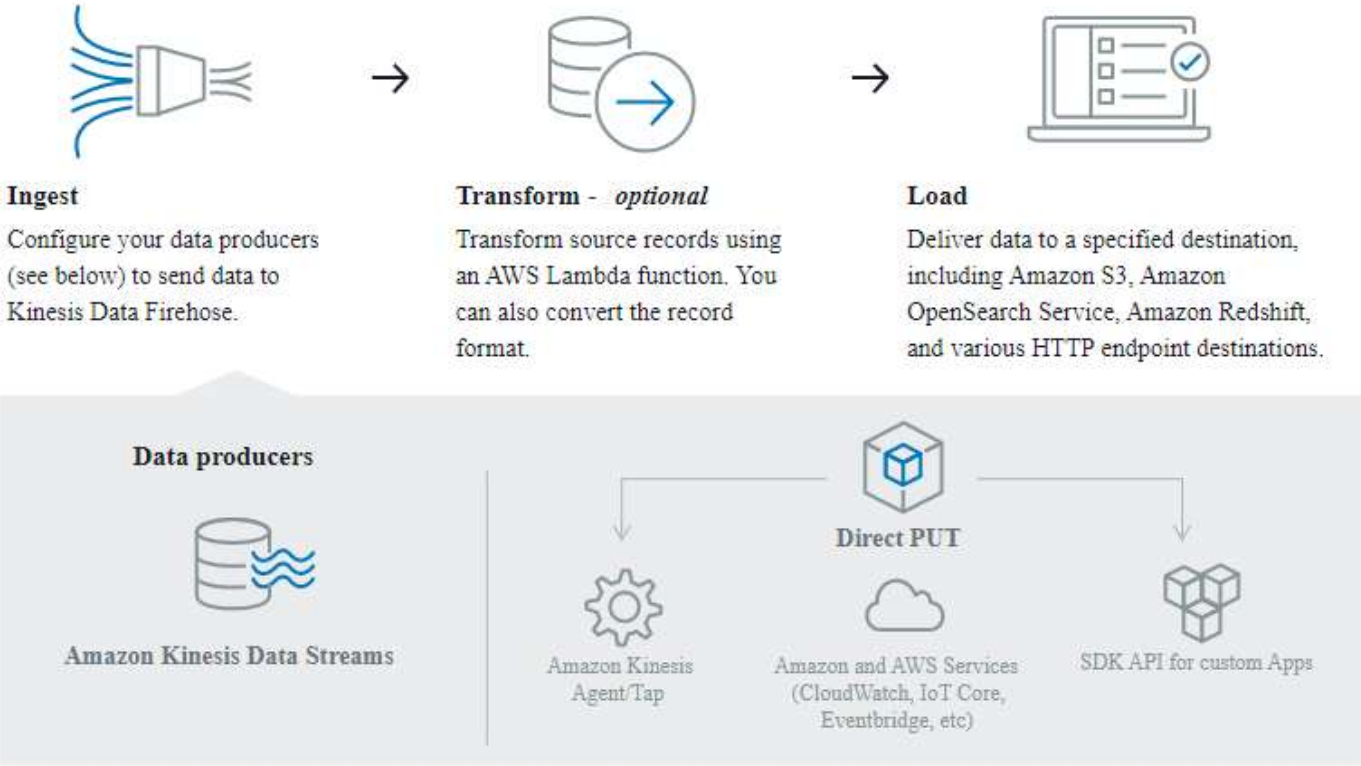
Kinesis Data Firehose

- Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers.
- With Kinesis Data Firehose, you don't need to write applications or manage resources. You configure your data producers to send data to Kinesis Data Firehose, and it automatically delivers the data to the destination that you specified.
- You can also configure Kinesis Data Firehose to transform your data before delivering it.

Kinesis Data Firehose – Key Concepts

- **Kinesis Data Firehose delivery stream**
  - The underlying entity of Kinesis Data Firehose. You use Kinesis Data Firehose by creating a Kinesis Data Firehose delivery stream and then sending data to it.
- **Record**
  - The data of interest that your data producer sends to a Kinesis Data Firehose delivery stream. A record can be as large as 1,000 KB.
- **Data Producer**
  - Producers send records to Kinesis Data Firehose delivery streams. For example, a web server that sends log data to a delivery stream is a data producer.
  - You can also configure your Kinesis Data Firehose delivery stream to automatically read data from an existing Kinesis data stream, and load it into destinations.
- **Buffer size and Buffer interval**
  - Firehose buffers incoming data to a certain size or for a certain period of time before delivering it to destinations. Buffer Size is in MBs and Buffer Interval is in seconds.

Kinesis Data Firehose – How it works?



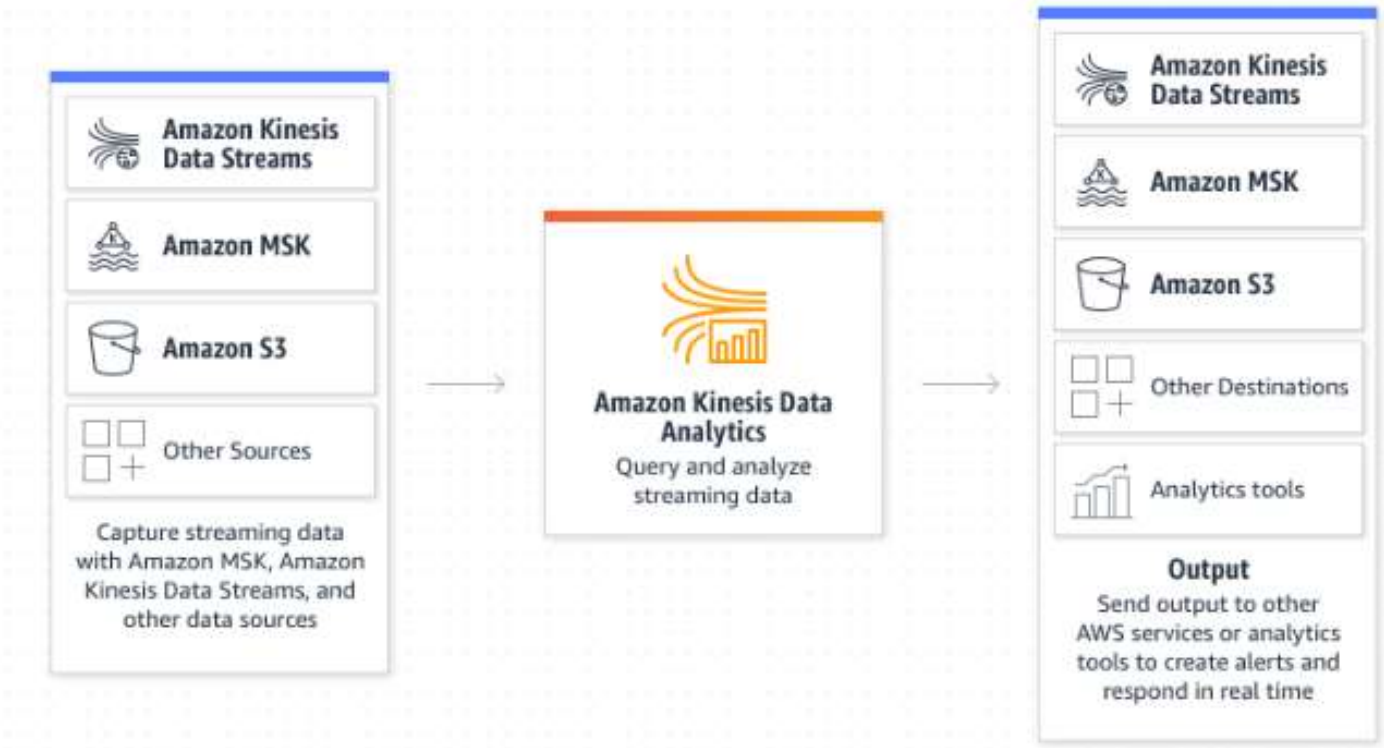
Kinesis Data Analytics

Kinesis Data Analytics allows you to transform and analyze streaming data in real time using Apache Flink.

With Kinesis Data Streams you can:

- Run your Apache Flink applications continuously and auto-scale with no setup cost and without managing servers.
- Process data with sub-second latencies from data sources like Amazon Kinesis Data Streams and Amazon MSK, and respond to events in real time.
- Analyze streaming data interactively using managed Apache Zeppelin notebooks with Kinesis Data Analytics Studio.
- Build applications in SQL, Java, Python, or Scala. Perform joins, filters, aggregations over time windows, and more.

Kinesis Data Analytics



Amazon Kinesis Demos

Lab 1 – Kinesis producers using SDK and KPL

Instructions Document: L01-Kinesis-Ingest-SDK-KPL.txt

In this lab, we perform the following operations:

- Create a Kinesis Data Stream
- Launch a Cloud9 instance and open the IDE environment
- Ingest records into the data stream using Python boto3 SDK
- Ingest records into the data stream using KPL

Lab 2 – Process data using Lambda and write to DynamoDB

Instructions Document: L04-Kinesis-Client-Lambda-DynamoDB.txt

In this lab, we perform the following operations:

- Create a Kinesis Data Stream
- Process the records of stream using Lambda
- Write the processed data to DynamoDB table

Lab 3 – Write stream data to S3 using Kinesis Data Firehose

Instructions Document: L06-Kinesis-Firehose-DataStream-to-S3.txt

In this lab, we perform the following operations:

- Setup a Firehose Delivery Stream to write records from Data Stream to S3
- Write files in S3 in parquet format

Lab 4 – Data pipeline using Kinesis Data Analytics

Instructions Document: L07-Kinesis-Data-Analytics.txt

In this lab, we perform the following operations:

- Demonstrate and end-to end data pipeline where we:
  - Produce data to Kinesis Data Stream using an SDK script
  - Perform real-time aggregations on the input stream and write the results to an output data stream using a Kinesis Data Analytics Studio Zeppelin notebook and Apache Flink.
  - Setup a Firehose Delivery Stream to deliver the data from the output stream to S3

Amazon OpenSearch

Formerly, Amazon ElasticSearch

Amazon OpenSearch

- **Amazon OpenSearch** is a **fully open-source search and analytics engine** for use cases such as:
  - log analytics
  - real-time application monitoring
  - clickstream analysis.

- **Amazon OpenSearch Service** is a managed service that makes it easy to deploy, operate, and scale OpenSearch clusters in the AWS Cloud.

  - Amazon OpenSearch Service supports OpenSearch and legacy ElasticSearch OSS (up to 7.10, the final open source version of the software).
  - When you create a cluster, you have the option of which search engine to use.

How OpenSearch works?

- **Amazon OpenSearch Service** provisions all the resources for OpenSearch cluster and launches it.
- It also automatically detects and replaces failed OpenSearch Service nodes, reducing the overhead associated with self-managed infrastructures. You can scale your cluster with a single API call or a few clicks in the console.

OpenSearch - Basic Terminology

- **Domain**
  - An OpenSearch Service domain is equivalent to an OpenSearch cluster.
  - Each EC2 instance in the cluster acts as one OpenSearch Service node.
- **Node**
  - An OpenSearch node is a single OpenSearch process.
  - The minimum number of nodes for a highly available OpenSearch cluster is three.
- **Document**
  - Basic unit of data in OpenSearch. (similar to a record in RDBMS)
  - Documents are stored in JSON format
- **Index**
  - An OpenSearch index is a collection of documents in OpenSearch.
  - Each index is split into shards.

OpenSearch - Basic Terminology

- **Shard**
  - OpenSearch shards enable parallelization of data processing across both a single and multiple OpenSearch nodes.
  - By default, OpenSearch automatically manages shard allocation within the node(s).

- **Replica**
  - OpenSearch replicas serve as a backup for shards and also aid in search performance by providing additional capacity.
  - OpenSearch automatically creates five primary shards and one replica for every index. You can add or remove replicas at any time to scale out query processing.

Amazon OpenSearch Labs

Lab 1 – OpenSearch basic operations

Instructions Document: L01-OpenSearch-Basics.txt

In this lab, we perform the following operations:

- Create and launch an OpenSearch domain
- Upload documents to an index using API (via cURL)
- Search for the documents using API
- Query the documents using OpenSearch Dashboards (formerly Kibana)
- Load sample data to the OpenSearch and create a fully interactive dashboard.

THANK YOU