What is cloud computing

- Cloud computing, often referred to as simply the cloud, is on-demand delivery of IT resources and applications via the Internet with pay-as-you -go pricing.
- With cloud computing, we don’t need to make large up-front investments in hardware and spend a lot of time managing that hardware. Instead, we can provision exactly the right type and size of computing resources we need.
- With cloud computing, we can access as many resources as we need, almost instantly, and only pay for what we use.
- Cloud computing provides an easy way to access servers, storage, databases, and a broad set of application services over the Internet.
- Cloud computing providers such as AWS own and maintain the network-connected hardware required for these application services, while we provision and use what we need for our workloads.

Advantages of Cloud Computing

- Low cost
- Usability
- Bandwidth
- Accessibility
- Disaster Recovery
- Secure

3 ways of cloud compute

3 building block of cloud computing are

- Saas
- Paas

- Iaas

Saas (Software as a service)

Software as a service is the easiest way to cloud compute. The software's are accessed over the internet.

Eg. Google doc, spreadsheet etc.

Advantage of saas is

- all application is free and paid via subscription.
- Accessible from any computer.

Paas (Platform as a service)

Provides environment and tool for creating new online applications

Eg. Google app engine, salesforce (force.com)

Advantages

- Facilitation of hosting capabilities.
- Designing and developing the application.
- Private or public deployment.

Disadvantage

- Application build on one vendor cannot be moved to another.

Iaas (Infrastructure as a Service)

Allow existing application to run on a cloud supplier's hardware. Iaas comes in 4 categories

1. Private cloud
2. Dedicated host
3. Hybrid hosting

4. Cloud hosting

   Aws works on iaas.

Advantages

- The application is hosted centrally
- Software testing takes place at a faster rate
- Reduction in IT operational cost.

Cloud types

- Public cloud
- Private cloud
- Hybrid cloud
- Community cloud

Public cloud

- Service providers use the internet to make resources, such as applications, storage
- Examples of public clouds include Amazon web service, IBM's Blue Cloud, Sun Cloud, Google compute engine and Windows Azure Services Platform.
- For users, these types of clouds will provide the best economies of scale, are inexpensive to set-up because hardware, application and bandwidth costs are covered by the provider. It's a pay-per-usage model and the only costs incurred are based on the capacity that is used.

Disadvantage

- the public cloud may not be the right fit for every organization. The

model can limit configuration, security, and SLA specificity, making it less-than-ideal for services using sensitive data that is subject to compliancy regulations

Private cloud

- Private clouds are data center architectures owned by a single company that provides flexibility, scalability, provisioning, automation and monitoring.
- The goal of a private cloud is not sell “as-a-service” offerings to external customers but instead to gain the benefits of cloud architecture without giving up the control of maintaining your own data center.
- Private clouds can be expensive, so most typically use by large enterprises. Private clouds are driven by concerns around security and compliance, and keeping assets within the firewall.

Hybrid cloud

- By using a Hybrid approach, companies can maintain control of an internally managed private cloud while relying on the public cloud as needed. For instance, during peak periods individual applications, or portions of applications can be migrated to the Public Cloud.
- This will also be beneficial during predictable outages: hurricane warnings, scheduled maintenance windows, rolling brown/blackouts.

Community cloud

- A community cloud is a multi-tenant infrastructure that is shared among several organizations from a specific group with common computing concerns.

Why AWS

- The free tier.
- On the go pricing.
- Performance.
- Deployment speed.
- Security.
- Flexibility.

AWS HISTORY

- Amazon is initially online retail seller.
- aws is launched in 2006.
- Amazon converts the unused storage infrastructure as business “Simple Storage web service” S3.
- By the end of 2006, Elastic Compute Cloud (EC2) was launched.
- Today AWS providing 70+ web services across 190 countries.
- Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.

AWS is located in 16 geographical "regions":

- North America (6 regions)
  - US East (Northern Virginia), where the majority of AWS servers are based
  - US East (Ohio)

    - US West (Oregon)
    - US West (Northern California)
    - AWS GovCloud (US), based in the Northwestern United States, provided for U.S. government customers, complementing existing government agencies already using the US East Region
    - Canada (Central)
- South America (1 region)
    - Brazil (São Paulo)
- Europe / Middle East / Africa (3 regions)
    - EU (Ireland)
    - EU (Frankfurt), Germany
    - EU (London), United Kingdom
- Asia Pacific (6 regions)
    - Asia Pacific (Tokyo), Japan
    - Asia Pacific (Seoul), South Korea
    - Asia Pacific (Singapore)
    - Asia Pacific (Mumbai), India
    - Asia Pacific (Sydney), Australia
    - China (Beijing)
        - **Region** is a distinct geographic location where amazon has its infrastructure
        - All the regions are designed to be independent of each other with separate power sources, internet connectivity and

geographic location

- An **availability zone** is a separate datacenter within a region. Amazon has intentionally kept region independent of each other if one goes down it does not have effect on other.
- For e.g. amazon have 2 AZ in Mumbai ap-south-1a, ap-south-1b.
- **Edge location** are cdn end points. edge locations are used by cloud front to cache files near the user who access them. For e.g. if a user wants to watch movie it's better to cache the movie to location near the user for latency
- Amazon cloud front and amazon route 53 are offered at edge location
- User can select the region depending upon following criteria
    1. User proximity – choose the base closer to the user
    2. Cost – cost may varies based on region
    3. Compliance – laws of lands such as data protection laws will influence your choice of regions.
    4. Service availability – not all services are available in a region

Understanding AWS Console

Aws console provides convenient access AWS services such as compute, storage and other cloud resources. Almost all web services are accessed at console.aws.amazon.com

AWS SERVICES

- **Storage**
    1. Amazon simple storage service (s3)
    2. Amazon glacier

3. Amazon elastic file system (EFS)
4. Amazon elastic block storage (EBS)

- **Compute**
    1. Elastic compute cloud (EC2)
    2. Amazon virtual private cloud (VPC)
    3. Auto scaling
- **Networking and content delivery**
    1. Route 53
    2. Elastic load balancer
    3. Cloudfront
- **Developers tool**
    1. Aws command line interface
- **Database**
    1. Amazon DynamoDB
    2. Amazon SimpleDB
    3. Amazon relationalDB
    4. Amazon aurora
- **Management tools**
    1. AWS CloudFormation
    2. AWS CloudWatch
- **Security, identity and compliance**
    1. Identity and access management (IAM)
- **Messaging**
    1. Amazon simple Queue Service(SQS)

2. Amazon simple notification service (SNS)
3. Amazon simple email service (SES)

IAM

- IAM (Identity and access management) allows you to manage users and their level of access to aws console. It provides multifactor authentication. Provides temporary access for users, services where necessary.
- It allows you to setup and maintain password rotation policy.
- Using IAM, organizations can create and manage AWS users and groups and use permissions to allow and deny their access to AWS resources.

Root User

(Note: When you first create an AWS account, you begin with only a single sign-in principal that has complete access to all AWS Cloud services and resources in the account. This principal is called the *root user)*

- The root user is similar in concept to the UNIX root or Windows Administrator account—it has full privileges to do anything in the account, including closing the account.
- The root user can be used for both console and programmatic access to AWS resources.

IAM Users

- IAM users is similar to normal users in Linux, this user can interact with the console and use the CLI.

Creating IAM USERS

Goto IAM -> users -> create users -> (name of the user) -> access type (AWS Management Console access) -> set password.

(Note: if u want to login then get the url from dashboard paste it and give the username and password)

(It is possible to customize the url in dashboard)

(SHOW: Delete User, Change user passwd , New Access key)

1.ARN (Amazon resource name is a unique name used for the identification of user or group, It is the combination of aws account id and user or group name)

- Max 5000 users in an aws account.

GROUPS

- A group is the collection of users having similar responsibility.
- You can use propagate permissions to users.
- Max 100 groups in an aws account.
- An IAM user can be member of 10 groups.

Creating group

Goto group -> create user -> done

ROLE

- An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS.
- instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

3 types of roles

1. Amazon service Roles—Granting permissions to applications running on an Amazon EC2 instance.
2. Cross-Account Access—Granting permissions to users from other AWS accounts, whether you control those accounts or not.
3. Identity provider access - Granting permissions to users authenticated by a trusted external system.

- Each role can have up to 10 policies attached.
- Max 500 roles in an aws account.

Policy

- A *policy* is a document that fully defines a set of permissions to access and manipulate AWS resources. Policy documents contain one or more permissions, with each permission defining:
    1. Effect—A single word: Allow or Deny
    2. Service—For what service does this permission apply? Most AWS Cloud services support granting access through IAM, including IAM itself.
    3. Resource—The resource value specifies the specific AWS infrastructure for which this permission applies. This is specified as an *Amazon Resource Name (ARN)*.

Identity Provider

- Identity providers are used to define trusted identity resources.
- With an identity provider (IdP), you can manage your user identities outside of AWS and give these external user identities permissions to use AWS resources in your

account. This is useful if your organization already has its own identity system, such as a corporate user directory. It is also useful if you are creating a mobile app or web application that requires access to AWS resources.

Multi-Factor Authentication (MFA)

- Multi-Factor Authentication (MFA) can add an extra layer of security to your infrastructure by adding a second method of authentication beyond just a password or access key. With MFA, authentication also requires entering a One-Time Password (OTP) from a small device. The MFA device can be either a small hardware device you carry with you (fot example SafeNet IDProve 100 (OTP Token)) or a virtual device via an app on your smart phone (for example google authentication)

  Add MFA: goto activate MFA for root account -> manage MFA -> virtual MFA -> copy the qr to google authenticator -> type 2 otp.

Password Policy

- A password policy is a set of rules that define the type of password an IAM user can set.

  (Note: Go through the options)

Credential Report

- It lists all your account's users and the status of their various credentials including passwords, access keys, and MFA devices.

Encryption keys

- AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data. AWS KMS is integrated with other AWS services including Amazon Elastic Block Store (Amazon EBS), Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Elastic Transcoder, Amazon WorkMail, Amazon Relational Database Service (Amazon RDS), and others to make it simple to encrypt your data with encryption keys that you manage
- AWS KMS lets you create master keys that can never be exported from the service and which can be used to encrypt and decrypt data based on policies you define.

Amazon Elastic Compute Cloud (Amazon EC2)

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud.
- Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster
- Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Features of Amazon EC2

- Virtual computing environments, known as *instances*
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types.*
- Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)

(create an ec2 instance and explain about instance family)

EC2 on demand

1. General purpose.
2. Compute optimized

    Compute intensive application.

3. Memory optimized

    Database and memory caching application.

4. Storage optimized

    Data warehousing and parallel computing.

EC2 Options

- **On-Demand instances**

    Pay for the instances that you use by the hour, with no long-term commitments or up-front payments.

- **Reserved Instances**

    Make a low, one-time, up-front payment for an instance, reserve it for a one- or three-year term, and pay a significantly lower hourly rate for these instances.

- **Spot instances**

    Bid on unused instances, which can run as long as they are available and your bid is above the Spot price, at a significant discount.

- **Dedicated hosts**

    Pay for a physical host that is fully dedicated to running

your instances, and bring your existing per-socket, per-core, or per-VM software licenses to reduce costs.

On demand vs Reserved vs Spot Instances

- On demand

  1.Users that want the low cost and flexibility of Amazon EC2 without any upfront payment or long term commitment.

  2. Application with short term, spiky, or unpredictable workloads that cannot be interrupted.
  3. Application which is developed or tested on Amazon EC2 for the first time.

  (note: Pay per hour)

- Reserved

  1. Application with steady state or predictable usage

     Eg; web servers running in an instance

  2. Application that require specific capacity
  3. Users are able to make upfront payment to reduce the total computing cost.

- Spot

  1. Application that has flexible start and end time.
  2. Application that are feasible at very low compute prices.
  3. Users with urgent computing needs for large amount of additional capacity

- Dedicated Host

  1. An Amazon EC2 Dedicated Host is a physical server with

EC2 instance capacity fully dedicated to your use.

2. Dedicated Hosts allow you to use your existing per-socket, per-core, or per-VM software licenses, including Windows Server, Microsoft SQL Server, SUSE and Linux Enterprise Server.

(create a windows instance -> go to connect option -> download the remote desktop file -> get password -> choose key pair -> get the password -> open remote desktop -> enter the password)

PRACTICAL

1. Login to an instance

   Using SSH if it is a Linux Instance.

   Using RDP (Remote desktop protocol) if it is a Windows Instance

2. Launch more like this

   Goto instance -> launch more like this

   (Note: This option does not clone your selected instance, it only replicates some configuration details.)

3. Termination protection

   Select the instance -> action ->instance setting -> change termination or you can add while creating a instance.

4. Attaching role

   Select the instance -> action ->instance setting ->attach/replace iam role

5. Change instance type

Stop the instance -> action -> instance setting -> change instance type

(note: When an instance terminates, the data on any instance store volumes associated with that instance is deleted.)

EBS

- Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances.
- EBS volumes are highly available and reliable storage volumes that can be attached to any running instance that is in the same Availability Zone.

STORAGE BACKED BY EBS

1. EBS General Purpose SSD (gp2)
2. Provisioned IOPS SSD (io1)
3. Throughput Optimized HDD (st1) (cant see for root add a new ebc volume and check,its not a boot volume)
4. Cold HDD (sc1)

- **General Purpose SSD** (gp2) volumes, you can expect base performance of 3 IOPS/GiB, with the ability to burst to 3,000 IOPS for extended periods of time. Gp2 volumes are ideal for a broad range of use cases such as boot volumes, small and medium-size databases, and development and test environments. Gp2 volumes support up to 10,000 IOPS and 160 MB/s of throughput.
- **Provisioned IOPS SSD** (io1) volumes, you can provision a specific level of I/O performance. Io1 volumes support up to 20,000 IOPS and 320 MB/s of throughput. This allows you to predictably scale to tens of thousands of IOPS per EC2 instance.

- **Throughput Optimized HDD** (st1) volumes provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. With throughput of up to 500 MiB/s, this volume type is a good fit for large, sequential workloads such as Amazon EMR, ETL, data warehouses, and log processing.
- **Cold HDD (sc1**) volumes provide low-cost magnetic storage that defines performance in terms of throughput rather than IOPS. With throughput of up to 250 MiB/s, sc1 is a good fit ideal for large, sequential, cold-data workloads. If you require infrequent access to your data and are looking to save costs, sc1 provides inexpensive block storage.

- EBS volumes are created in a specific Availability Zone, and can then be attached to any instances in that same Availability Zone
- You can mount multiple volumes on the same instance, but each volume can be attached to only one instance at a time.
- To make a volume available outside of the Availability Zone, you can create a **snapshot** and restore that snapshot to a new volume anywhere in that region. You can copy snapshots to other regions and then restore them to new volumes there, making it easier to leverage multiple AWS regions for geographical expansion, data center migration, and disaster recovery.

  (Note: Create an instance -> attach a new volume to that instance -> mount the volume -> add some files -> detach -> attach the volume to another instance -> verify the contents are present)

PRACTICAL

1. Attaching the volume to different instance

add a new volume to an existing instance -> mount it -> add some content -> unmount -> detach the volume -> attach the volume to another instance -> mount and verify

(do not **detach** the root volume)

2. It is possible to extend the size of a volume

Select the volume-> action -> modify volume (it may take some time to take effect)

Snapshot

- You can back up the data on your EBS volumes to Amazon S3 by taking point-in-time snapshots.
- Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved
- For creating consistent snapshot stop the volume and take the snapshot, snapshot can be taken without stopping the volume but may be inconsistent.
- All snapshots are stored in s3 service of aws.
- These snapshots can be used to create multiple aws volume across availability zone.
- Snapshots can be shared with specific aws accounts or made public.

PRACTICAL

1. Create snapshot from a volume

Stop the instance -> goto volume -> action ->create snapshot

2. Create volume from a snapshot

   Select the snapshot -> action -> create volume -> select the desired availability song.

3. Copying snapshot

   Snapshot can be copied to different region, copying in another region helps to create a volume in that region

   Select the snapshot -> action -> copy -> select the desired region

4. Sharing snapshot

   Snapshot can be shared between the users or make it private.

   Select the snapshot -> action -> modify permission -> public

5. Delete snapshot

   Select a snapshot -> action -> delete

AMI

- An Amazon Machine Image (AMI) is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud ("EC2"). It serves as the basic unit of deployment for services delivered using EC2.

Launch Permissions

- The owner of an AMI determines its availability by specifying launch permissions. Launch permissions fall into the following categories.

1. Public -> The owner grants launch permissions to all AWS accounts.
2. Explicit -> The owner grants launch permissions to specific AWS accounts.
3. Implicit -> The owner has implicit launch permissions for an AMI.

PRACTICAL

1. Creating ami

   Create and instance with webpage -> create snap from volume -> create image from snap -> launch

2. Launch permission

   Select the ami -> modify image permission

3. Copying the ami

   Select the ami -> copy. (while copying the ami the snapshot will be copied too destination)

SECURITY GROUPS

- A ***security group*** acts as a virtual firewall that controls the traffic for one or more instances.
- When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances.

- the new rules are automatically applied to all instances that are associated with the security group.
- Max 500 sg.
- Max 100 rules for a sg (50 inbound and 50 outbound)
- For each rule, you specify the following.

    1. Type: protocol
    2. Protocol: The protocol to allow.
    3. Port range: For TCP, UDP, or a custom protocol, the range of ports to allow. You can specify a single port number (for example, 22), or range of port numbers (for example, 7000-8000).
    4. Source or destination: The source (inbound rules) or destination (outbound rules) for the traffic

Amazon EC2 Key Pairs

- Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. Public–key cryptography uses a public key to encrypt a piece of data, such as a password, then the recipient uses the private key to decrypt the data. The public and private keys are known as a *key pair.*
- To log in to your instance, you must create a key pair, specify the name of the key pair when you launch the instance, and provide the private key when you connect to the instance. Linux instances have no password, and you use a key pair to log in using SSH. With Windows instances, you use a key pair to obtain the administrator password and then log in using RDP.
- Max 5000 keypair.

PRACTICAL

1. Deleting a key

Goto keypair -> select the private key you want to delete

2. Import key

   Open puttygen -> create public key -> import {wot user have is private key and what aws have is public key)

Elastic ip

- An **Elastic IP address** is a static IPv4 address designed for dynamic cloud computing.
- With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
- To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface.
- When you associate an Elastic IP address with an instance or its primary network interface, the instance's public IPv4 address (if it had one) is released back into Amazon's pool of public IPv4 addresses. You cannot reuse a public IPv4 address.
- A disassociated Elastic IP address remains allocated to your account until you explicitly release it.
- If an Elastic IP address is not associated with a running instance, or if it is associated with a stopped instance or an unattached network interface it will be charged.
- An Elastic IP address is region specific.
- When you associate an Elastic IP address with an instance that previously had a public IPv4 address, the public DNS hostname of the instance changes to match the Elastic IP address.
- 5 Elastic IP addresses per region.

Practical

1. Associate ip

   Goto elastic ip -> allocate a new address -> allocate -> actions -> associate address -> select instance

2. Dissociate ip

   Select the instance -> dissociate ip

3. Release ip

   Select ip -> release (only after dissociate we can release)

Network interfaces

- An elastic network interface (*network interface)* is a virtual network interface that you can attach to an instance in a VPC. Network interfaces are available only for instances running in a VPC.
- A network interface can include the following attributes:

  1. A primary private IPv4 address.
  2. One or more secondary private IPv4 addresses
  3. One Elastic IP address (IPv4) per private IPv4 address
  4. One public IPv4 address
  5. One or more IPv6 addresses

- You can create a network interface, attach it to an instance, detach it from an instance, and attach it to another instance.
- When you move a network interface from one instance to another, network traffic is redirected to the new instance.
- Every instance in a VPC has a default network interface, called the *primary network interface* (eth0). You cannot detach a primary network interface from an instance. You can create and attach additional network interfaces

(The maximum number of network interfaces that you can use varies by instance type).

Practical

1. Creating new network interfaces

   Create network interfaces -> add description -> subnet -> private ip (default) -> security group

AWS CLI

- The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

  ### Practical

  1. Attaching s3 role to ec2

     Create a role so that from ec2 it is possible to access s3 -> Launch and aws ami (no other ami can access s3 by default) -> select the role while creating -> launch the instance

  2. Help

     # aws

     # aws s3 help

  3. Accessing s3 from ec2

     # aws s3 ls

Cmd to list all buckets in s3

4. Creating bucket

   #aws s3 mb s3://<bucket name>

   Cmd to create bucket without specifying a region

   #aws s3api create-bucket --bucket <bucket name> --region us-east-1 (cannot create in all region because of endpoint)

5. Copying to bucket

   Bucket must a permission so that we can write the changes to bucket

   # aws s3 cp <F.N> s3://<bucket name>

6. Copying from bucket

   # aws s3 s3://<bucket name>/<F.N>.

7. Syncing Bucket with local disk

   #aws s3 sync s3://<bucket name>.

   Cmd to sync all data from bucket to local storage

8. Deleting a bucket

   #aws s3 rb s3://<bucket name>

   # aws s3 rb s3://<bucket name> --force

   Cmd to remove a non-empty bucket

#aws s3api delete-bucket --bucket <bucket name>

(note: Mumbai region buckets are not supported in awscli)

AWSCLI in RHEL

1. Install the python
2. Install pip (By default the package does not comes with repository, download and install python-pip from rpmfind.net, #yum localinstall python-pip)

   (pip is a package management system used to install and manage software packages written in Python)

3. Check pip is installed or not (#pip list)
4. Use pip to install awscli (#pip install awscli botocore, botocore is the package awscli uses for all its work)
5. Use aws cmd (#aws s3 ls

AWSCLI in WINDOWS

1. Launch an windows instance
2. Login
3. Install awscli msi installer from aws website (google for awscli for windows)
4. Open the cmd prompt and start executing aws cmd.

BOOTSTRAP SCRIPTS

- Bootstrap scripts are used for executing the set of cmds or scripts to run as soon as ec2-instance goes live in root level.
- We can do automation by using the scripts

Practical

Create an instance -> redhat ami -> advanced detail -> as a text ->

#!/bin/bash

yum install -y httpd

systemctl restart httpd

systemctl enable httpd

echo “checking for bootstrap script” > /var/www/html/index.html

systemctl restart httpd

systemctl enable httpd

->add SG (ssh and http) -> launch -> wait till status check is 2/2.

LOAD BALANCER

- Elastic Load Balancing distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications.

- The load balancer serves as a single point of contact for clients, which increases the availability of your application. You can add and remove instances from your load balancer.

Practical

1. Creating a load balancer

   Create an instance (rhel) -> install httpd -> service restart -> create index.html (any content) -> service -> add http for sg -> check it is working -> go to load balancer -> based on the load balancer you need select the load balancer -> classic load balancer -> add a name and default vpc -> select sg which support ssh and http -> configure health check -> response timeout (5 sec : time to wait when receiving a response from the health check) -> interval (amount of time between health checks) -> unhealthy threshold (2 no of consecutive health check failures before declaring an EC2 instance unhealthy, note : in 60 sec it checks 2 time since interval is 30) -> healthy threshold (no of consecutive health check successes before declaring an ec2 instance healthy) -> add the instance -> enable cross end load balancing (cross end load balancing distributes traffic evenly across all your back-end instances in all available zones) -> enable connection draining (the no.of sec to allow existing traffic to continue flowing) -> create -> wait for 1 min till it become in-service (status of instance) -> get the public dns of load balancer and paste it in the new tab e.g. dns/index.html

- When you create a load balancer in a VPC, you must choose whether to make it an internal load balancer or an Internet-facing load balancer.
- The nodes of an Internet-facing load balancer have public IP addresses. The DNS name of an Internet-facing load balancer is

publicly resolvable to the public IP addresses of the nodes. Therefore, Internet-facing load balancers can route requests from clients over the Internet.

- The nodes of an internal load balancer have only private IP addresses. The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes. Therefore, internal load balancers can only route requests from clients with access to the VPC for the load balancer.
- Max 20 load balancer per region.
- Max 5 SG for load balancer.
- Max 1 subnet for load balancer.
- Deleting a load balancer does not affect its EC2 instance.

PLACEMENT GROUPS (ONLY THEORY)

- A *placement group* is a logical grouping of instances within a single Availability Zone.
- Placement groups are recommended for applications that benefit from low network latency, high network throughput, or both.
- If you stop an instance in a placement group and then start it again, it still runs in the placement group. However, the start fails if there isn't enough capacity for the instance.
- A placement group can't span multiple Availability Zones.
- The name you specify for a placement group must be unique within your AWS account.

(note: placement group concept is mainly used in 10G network (for connecting DB , 10 gigabits per second (or 10 billion bits) check the instance , its available within an AZ because it’s not possible to connect from one AZ to another AZ using this connection).

AUTO SCALING

- Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.
- Collection of ec2 instance is called auto scaling groups.
- We can specify the minimum and maximum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes below or above this size.
- When you create an Auto Scaling group, you must specify a launch configuration.
- Max 20 auto scaling groups per region.

Practical

1. Creating a Launch Configuration Group

   A **launch configuration** is a template that an Auto Scaling group uses to launch EC2 instances. When we create a launch configuration, we specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping. If we launched an EC2 instance before, we can specify the same information in order to launch the instance.

   Create launch configuration -> select the instance -> assign a name and role if any -> add storage -> configure SG so that it can access http and ssh -> assign a key pair -> launch

2. Creating an Auto Scaling Group

Create an auto scaling group with existing launch configuration -> assign a group name -> group size with 1 instance -> select the default network -> add subnet available in region (subnet represent AZ, if we select only 1 subnet instance will be created in that AZ for fault tolerance create in all available AZ) -> use scaling policies to adjust the capacity of this group -> increase group size -> add a new alarm (give your mail id so that we will receive mail if cpu utilization reaches above the limit) -> ls : 60% -> period : 1min -> take the action add 1 instance -> decrease group size -> add a new alarm (give your mail id so that we will receive mail if cpu utilization reaches above the limit) -> ls : 30% -> period : 1min -> take the action remove 1 instance -> configure notification -> add tags -> create auto scaling group.

(note: for verification login to instance and use #yes > /dev/null & or #dd /dev/null > /dev/null).

CloudWatch

- Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time.
- We can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.
- CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define.
- 2 types
    1. Basic Cloudwatch – Monitors in every 5 min

2. Detailed Cloudwatch - Monitors in every 1 min

- Basic monitor matrices are free for EC2, EBS, ELB, RDS.
- By default, basic CloudWatch is enabled.

Practical

1. Detailed CloudWatch

We can enable detailed CloudWatch either while creating an instance or clicking on action -> CloudWatch -> enable detailed monitoring

2. Create an alarm

Cloudwatch -> create an alarm -> ec2 metrics -> per-instance metrics -> select an alarm (cpuutilization) -> name (highloadavg) -> description (high load avg) -> whenever (cpu utilization is more than 80%) -> actions -> whenever this alarm (state is ALARM) -> send notification to (loadavg) -> +EC2 option (optional option) -> take this action (stop this instance) -> period (if we want to change).

or

select the instance -> action -> cloudwatch -> edit/add alarm -> create a alarm -> shut down the instance when reaching cpuutilization 60% -> save.

CREATE A BILLING CYCLE

- We can monitor our AWS costs by using CloudWatch. With CloudWatch, you can create billing alerts that notify us when our usage of services exceeds thresholds that we define.
- We can specify these threshold amounts when we create the

billing alerts.

- When our usage exceeds these amounts, AWS sends us an email notification.
- We can also sign up to receive notifications when AWS prices change.

PRACTICAL

Goto my billing dashboard -> alerts $ notification -> receive billing alert -> manage billing alert (it will take you to cloudwatch in N.Virginia) -> click on billing -> create alarm -> exceed ($10) -> send a notification to -> new list -> give a mail id -> create an alarm -> check mail for confirmation.

S3

- Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web.
- S3 is object based i.e. allows you to upload files.
- Files are stored in bucket.
- A bucket is a logical unit of storage used to store data in S3. Buckets have a unique namespace for each region.
- S3 is region specific i.e. data is stored in both the AZ of the region.
- It is designed to deliver 99.999999999% durability.
- Amazon S3 supports data transfer over SSL and automatic encryption of your data once it is uploaded.
- Files can be from 1 byte to 5 tb.
- By default, you can create up to 100 buckets in each of your AWS accounts.
- A bucket has no size limit. It can store numbers of objects of any

size

Free Usage

1. Free 5GB usage storage
2. 20,000 gets
3. 2000 puts
4. 15GB data transfer

   (All are monthly basis)

Storage type

1. Standard s3 storage
2. Standard s3 - Infrequent Access
3. Reduced redundancy
4. Amazon glacier

- **Standard s3 storage:** This storage class is ideal for performance-sensitive use cases and frequently accessed data. It is the default storage class; if you don't specify storage class at the time that you upload an object, Amazon S3 assumes the standard storage class.
- **Standard s3 - Infrequent Access** (**Standard - IA**): This storage class (IA, for infrequent access) is optimized for long-lived and less frequently accessed data, for example backups and older data where of access has diminished, but the use case still demands high performance.
- **Reduced redundancy:** The Reduced Redundancy Storage (RRS)

storage class is designed for noncritical, reproducible data stored at lower levels of redundancy than the STANDARD storage class, which reduces storage costs. The durability level corresponds to an average annual expected loss of 0.01% of objects. For example, if you store 10,000 objects you may loss 100 files.

- **Amazon glacier:** The GLACIER storage class is suitable for archiving data where data access is infrequent. Archived objects are not available for real-time access. You must first restore the objects before you can access them. The GLACIER storage class uses the very low-cost Amazon Glacier storage service.

(note: initially you might upload objects using the STANDARD storage class, and then use a bucket lifecycle configuration rule to transition objects STANDARD_IA or GLACIER storage)

(note: consistency model s3 uses read-after-write consistency for PUTS of new objects and eventual consistency for overwrite PUTS and DELETES)

Creating a bucket

1. Creating a bucket

   Create bucket -> select a unique name -> select the region you want to create bucket

2. Create a folder

   Select the bucket -> create a folder

3. Adding an object

   Select the bucket -> upload the file

4. Make public

   Select the file -> properties -> make public

Permission

- Bucket permissions specify who is allowed access to the objects in a bucket and what permissions you have granted them.
- You can grant the permission for:

  1. **Everyone**—Use this group to grant anonymous access
  2. **Authenticated Users**—This group consists of any user that has an Amazon AWS Account. When you grant the Authenticated User group permission, any valid signed request can perform the appropriate action. The request can be signed by either an AWS Account or IAM User.

  3. **Log Delivery**—This group grants write access to your bucket when the bucket is used to store server access logs.
  4. **Me**—This group refers to your AWS root account, and not an IAM user.

S3 VERSIONING

- Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures.

- Once we enable version in a bucket, it can never return to an unversioned state. You can, however, suspend versioning on that bucket.

Practical

(note: Before enabling version to create a file f1 with any content (hi) -> upload it -> then remove the content and add another content (hello) -> u can see the old content is replaced with the new content)

1. Enable versioning

   Select the bucket -> enable versioning

(note: do the same process as above in version tab go for show you can see all the version of file)

(note: it all also possible to recover the deleted file go to version tab -> show -> delete the file having the name “delete marker”)

2. Suspending versioning

   Select the bucket -> suspend versioning

Lifecycle Management

- You can manage the lifecycle of objects by using Lifecycle rules.
- Lifecycle rules enable you to automatically transition objects to the Standard - Infrequent Access Storage Class, and/or archive objects to the Glacier Storage Class, and/or remove objects after a specified time period.

Practical

1. Adding a Lifecycle rule

   Create a bucket -> enable version -> goto lifecycle -> add rule -> add rule either to bucket or to the directory in a bucket -> actions on current version -> select the option you want to enable -> actions on previous version (for versioning, what to do for file already existing) -> review -> create and save the rule.

   (note: you can directly upload files to **standard storage, standard storage -IA, reduced redundancy,** while selecting the file for uploading select set details -> select storage type).

Restoring from glacier

- Objects archived to Amazon Glacier are not accessible in real-time.
- We must first initiate a restore request and then wait until a temporary copy of the object is available for the duration (number of days) that you specify in the request.

- Amazon S3 processes only one restore request at a time per object.
- Temporary object is copied to RRS storage, so we have to pay for glacier as well as RRS.

Practical

Select the object -> initiate restore -> Specify the number of days that your archived data will be accessible (5, so data will be able to access temporarily) -> select retrieval option -> restore

Static website hosting

- You can host a static website on Amazon S3. On a static website, individual web pages include static content.
- To host your static website, you configure an Amazon S3 bucket for website hosting and then upload your website content to the bucket.

Practical

1. Creating static website

Create a bucket -> add the files (index.html and error page) -> make public -> give permission for bucket to everyone -> click on enable website hosting -> index document -> index.html -> error document (used to show custom error messages rather than unreachable error) -> error.html (incorrect website) -> click on endpoint to view

(note: if we enter incorrect url it will display the message

from error page)

2. Redirect all request to another host

   Redirects all request to -> enter the domain (here just give gmail.com to show how redirection works)

Logging

- In order to track requests for access to your bucket, you can enable access logging.
- Each access log record provides details about bucket name, request time, request action, response status, and error code, if any.
- Access log information can be useful in security and access audits.
- Logging is region specific.

Practical

1. Enable logging

   Select bucket -> give permission to log delivery -> logging -> enabled -> target bucket (bucket name where you want Amazon S3 to save the access logs as objects) -> target prefix (log file name).

Cross-region replication

- Cross-region replication is a bucket-level feature that enables

automatic, asynchronous copying of objects across buckets in different AWS regions.

- The object replicas in the destination bucket are exact replicas of the objects in the source bucket. They have the same key names and the same metadata.
- Existing objects of source bucket will not be copied to destination bucket.
- The source and destination buckets must be versioning-enabled.
- The source and destination buckets must be in different AWS regions.
- You can replicate objects from a source bucket to only one destination bucket.

Practical

1. Create cross region replication

   Select the bucket -> enable versioning -> enable cross region replication -> source bucket -> destination region (Oregon) -> destination bucket (give any name) -> Destination storage class (any) -> create an iam role -> save

   Verify by uploading a file in source bucket.

   (note: verify **logs** after cross region replication)

S3 Multipart Upload

- S3 multipart allows you to upload a single object in multiple part. The object is assembled after all uploads.
- Parts can be uploaded in parallel for high throughput.
- Uploads can be paused and resumed.

- Objects can be uploaded and while we are creating it.

S3 Data Encryption

- S3 data encryption provides added security for your data.
- Server-side encryption encrypts your data before storing it in its data center and decrypts it when you access it.
- S3 uses 256-bit Advanced Encryption Standard (AES) to encrypt your data.

Events

- The Amazon S3 notification feature enables you to receive notifications when certain events happen in your bucket.
- Events are

    1. A new object created event
    2. An object removal event
    3. A Reduced Redundancy Storage (RRS) object lost event

Tags

- Tags are used to identify and categories your aws resources.
- We can use tags to organize your AWS bill to reflect your own cost structure.
- Tags consists of key and value.

(note: mainly used to identify from which bucket bill is high)

Requester Pays bucket

- In general, bucket owners pay for all Amazon S3 storage and data transfer costs associated with their bucket.
- With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing

data.
- We can configure buckets to be Requester Pays when you want to share data but not incur charges associated with others accessing the data.

Amazon S3 Transfer Acceleration

- Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket
- Transfer Acceleration takes advantage of Amazon CloudFront's globally distributed edge locations. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path.
- When using Transfer Acceleration, additional data transfer charges may apply.

Use

- customers that upload to a centralized bucket from all over the world.
- transfer gigabytes to terabytes of data on a regular basis across continents.
- underutilize the available bandwidth over the Internet when uploading to Amazon S3

Storage Management

- Amazon S3 Storage Management capabilities helps you better analyze and manage your storage by

1. S3 Object Tagging
2. S3 Analytics, Storage Class Analysis
3. S3 Inventory
4. S3 CloudWatch Metrics

- **S3 Object Tagging** – With S3 Object Tagging you can manage and control access for Amazon S3 objects. S3 Object Tags are key-value pairs applied to S3 objects which can be created, updated or deleted at any time during the lifetime of the object. With these, you'll have the ability to create Identity and Access Management (IAM) policies, setup S3 Lifecycle policies, and customize storage metrics. These object-level tags can then manage transitions between storage classes and expire objects in the background.

- **S3 Analytics, Storage Class** Analysis – With storage class analysis, you can analyze storage access patterns and transition the right data to the right storage class. This new S3 Analytics feature automatically identifies the optimal lifecycle policy to transition less frequently accessed storage to SIA. You can configure a storage class analysis policy to monitor an entire bucket, a prefix, or object tag. Once an infrequent access pattern is observed, you can easily create a new lifecycle age policy based on the results. Storage class analysis also provides daily visualizations of your storage usage in the AWS Management Console. You can export these to an S3 bucket to analyze using the business intelligence tools of your choice, such as Amazon QuickSight.
- **S3 Inventory** – You can simplify and speed up business workflows and big data jobs using S3 Inventory, which provides a scheduled alternative to Amazon S3's synchronous List API. S3 Inventory provides a CSV (Comma Separated Values) flat-file output of your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or a shared prefix.
- **S3 CloudWatch Metrics** – Understand and improve the performance of your applications that use Amazon S3 by monitoring and alarming on 13 new S3 CloudWatch Metrics. You can receive 1-minute CloudWatch Metrics, set CloudWatch alarms, and access CloudWatch dashboards to view real-time operations and performance such as bytes downloaded and the

4xx HTTP response count of your Amazon S3 storage. For web and mobile applications that depend on cloud storage, these let you quickly identify and act on operational issues. By default, 1-minute metrics are available at the S3 bucket level. You also have the flexibility to define a filter for the metrics collected using a shared prefix or object tag, allowing you to align metrics to specific business applications, workflows, or internal organizations.

(**Note:** *Amazon S3* browser is a windows client tool to manage S3.)

PRACTICAL

(create a bucket -> permission everyone -> upload a small video -> public)

1. Playing the video from s3 using WordPress

   Launch an WordPress instance -> from syslog of instance login to WordPress website with the help of public ip (username: user, password: (get from syslog) -> click on post ->add new post -> paste the video link from s3

2. Playing the video from s3 using File

   Create a file add the following content

   ```
   <h1>From S3</h1>

   <video width="320" height="240" controls>

   <source src="https://s3.ap-south-1.amazonaws.com/awscdncheck/videoplayback.mp4" type="video/mp4">

   </video>
   ```

CDN

- A content delivery network or content distribution network (**CDN**) is a system of distributed servers that deliver webpages and other web contents to user based on geographic locations of the user, the origin of the webpage and content delivery server
- The goal of a **CDN** is to serve content to end-users with high availability and high performance.

Amazon CloudFront

- Cloudfront is amazon cdn.
- Amazon CloudFront is a global content delivery network (CDN) service that accelerates delivery of your websites, APIs, video content or other web assets through CDN caching.
- It integrates with other Amazon Web Services products such as S3, ec2, ELB, Route 53 to give developers and businesses an easy way to accelerate content to end users with no minimum usage commitments.
- CloudFront delivers your content through a worldwide network of data centers called edge locations.
- When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency, so that content is delivered with the best possible performance. If the content is already in the edge location with the lowest latency, CloudFront delivers it

immediately. If the content is not in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server.

CloudFront distributions

1. Web distribution – Uses the protocol HTTP or HTTPS to distribute media content
2. RTMP distribution – An RTMP (Real-Time Messaging Protocol) distribution allows an end user to begin playing a media file before the file has finished downloading from a CloudFront edge location.

Terms

1. **Origin Domain Name -** The DNS domain name of the Amazon S3 bucket or HTTP server from which you want CloudFront to get objects for this origin
2. **Origin Path -** If you want CloudFront to request your content from a directory in your Amazon S3 bucket or your custom origin, enter the directory path, beginning with a /. CloudFront appends the directory path to the value of Origin Domain Name.
3. **Origin ID** - A string that uniquely distinguishes this origin from other origins in this distribution.
4. **Restrict Bucket Access (Amazon S3 Only)**-Choose **Yes** if you want to require users to access objects in an Amazon S3 bucket by using only CloudFront URLs, not by using Amazon S3 URLs. Then specify the applicable values. Choose **No** if you want users to be able to access objects using either CloudFront URLs or Amazon S3 URLs.
5. **Viewer protocol policy** - Choose the protocol policy that you want viewers to use to access your content in

CloudFront edge locations.

6. **Allowed http methods** - Specify the HTTP methods that you want CloudFront to process and forward to your origin:

   (note: GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE: You can use CloudFront to get, add, update, and delete objects, and to get object headers. In addition, you can perform other POST operations such as submitting data from a web form)

7. **Cached HTTP method** - Specify whether you want CloudFront to cache the response from your origin when a viewer submits an OPTIONS request. CloudFront always caches the response to GET and HEAD requests.
8. **Forward header** - Specify whether you want CloudFront to forward request headers to your origin server and to cache objects based on header values.
9. **Object caching** – Specify how long the objects stay in the CloudFront cache.
10. **Minimum TTL -** Specify the minimum amount of time, in seconds, that you want objects to stay in CloudFront caches. The default value for **Minimum TTL** is 0 seconds.
11. **Forward Cookies** - Specify whether you want CloudFront to forward cookies to your origin server.
12. **Query String Forwarding and Caching -** CloudFront can cache different versions of your content based on the values of query string parameters.
13. **Smooth Streaming -** Choose **Yes** if you want to distribute media files in the Microsoft Smooth Streaming format using the origin that is associated with this cache behavior. Otherwise, choose **No.**
14. **Restrict Viewer Access -** If you want requests for

objects that match the PathPattern for this cache behavior to use public URLs, choose **No** Else **Yes.**

15. **SSL Certificate –** Specifies the option to access your domain by using defaults cloudfront or domain or custom cloudfront domain.

PRACTICAL

1. Creating CloudFront

cloudfront -> create distribution -> web -> select origin domain name (bucket dns name) -> viewer protocol policy (HTTP & HTTPS) -> allowed HTTP method (GET, HEAD) -> price class (use all edge location) -> create distribution.

Copy the domain name from distribution and replace it in the file -> wait till the cdn gets deployed (upto 20 mins)

2. Creating a CloudFront with BITNAMI

Use the existing WordPress instance that we have launched -> create post paste the cdn url.

(note: For cdn replace the cdn domain name with s3 and https with http)

3. **Creating a CloudFront with File**

Use the existing file add

<h1>From CDN</h1>

<video width="320" height="240" controls>

<source src="http://s3.ap-south-1.amazonaws.com/awscdncheck/videoplayback.mp4" type="video/mp4">

</video>

(note: to verify video is public)

4. **Create private content**

- Mainly used to securely serve this private content using CloudFront
- users access your private content by using special CloudFront signed URLs or signed cookies.
- users access your Amazon S3 content using CloudFront URLs, not Amazon S3 URLs.

Cloudfront -> private content -> origin access identity (oai) -> create oai (CloudFront OAI to objects S3 bucket) -> select the existing distribution -> goto origin -> edit -> restrict bucket access (yes) -> origin access identity (use an existing one) -> your identities (CloudFront OAI to objects S3 bucket) -> grant read permission on bucket (yes update bucket policy) ->verify the

bucket policy is update by selecting the bucket click on edit bucket policy there you can see the updated policy -> select the object in the bucket -> remove “everyone” permission to restrict public access from s3 -> save -> check the file with s3 link which won’t be able to access.

Origin access identity

- An origin access identity is a special CloudFront user that you can use to give CloudFront access to your Amazon S3 bucket.
- This is useful when you are using signed URLs or signed cookies to restrict access to private content in Amazon S3.

EFS

- Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2.
- With EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files, so your applications have the storage they need, when they need it.
- Multiple Amazon EC2 instances can access an EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance or server.
- With Amazon EFS, you pay only for the storage used by your file system.
- You don't need to provision storage in advance and there is no minimum fee or setup cost.
- Amazon EFS uses the protocol V4 and V4.1 to share the file system.

- Amazon EFS file systems store data and metadata across multiple Availability Zones in a region and can grow to petabyte scale, drive high levels of throughput, and allow massively parallel access from Amazon EC2 instances to your data.

  (Note: Amazon EFS is a file storage service for use with Amazon EC2. Amazon EFS provides a file system interface, file system access semantics (such as strong consistency and file locking), and concurrently-accessible storage for up to thousands of Amazon EC2 instances whereas Amazon S3 is an object storage service. Amazon S3 makes data available through an Internet API that can be accessed anywhere. We can use any az for mounting. Max 128 active connection at same time)

Practical

1. Creating and accessing EFS

   Create a sg (efs) -> add nfs as inbound rule -> goto efs -> create a file system -> select the default vpc -> remove all sg and add efs as the sg -> next ->create file system -> launch an redhat instance under efs sg -> login

   $sudo su

   #yum update -y && yum install -y nfs-utils

   #systemctl restart nfs-server

   #systemctl enable nfs-server

   #mkdir /efs

   #mount -t nfs4 <ip of efs>:/ /efs (check the subnet of instance, then in efs check the ip of corresponding subnet)

#df -h

STORAGE GATEWAY

- AWS Storage Gateway is a service that connects an on-premises software appliance with cloud-based storage to provide seamless and secure integration between your on-premises IT environment and the AWS storage infrastructure.
- The service enables you to securely store data in the AWS Cloud for scalable and cost-effective storage.

(Basically, it is an application that we install on vSphere or Hyper-V and associate with our aws account, this tool will asynchronously copy your data to s3)

- 4 types of Storage Gateway
  1. File gateway
  2. Volume gateway
  3. Tape gateway

✓ File gateway

a. Files are stored as object in s3 bucket and allows you to store and retrieve objects through nfs.

b. Once the object is transferred to s3 they can be managed as native s3 object so we can apply bucket polices such as versioning, lifecycle

management, cross region replication can be applied directly.

(it is mainly used to copy your files to s3, the files can be videos, images, documents etc.)

(uses nfs v3 or 4.1)

- ✓ Volume gateway
    a. It is a block based storage which uses iscsi block protocol mainly used to store os, applications, db etc.
    b. It acts as a virtual hard disk in cloud.

        (Basically, we are taking the hard disk on premise and we back them up as a virtual hdd in cloud)
    c. Volume gateway are 2 different type
        1. Stored volume: we can configure to store the primary data locally and then asynchronously back up point-in-time snapshots of this data to Amazon S3.
        2. Cached volume: we store our data in S3 and retain a copy of frequently accessed data locally. Cached volumes offer a substantial cost savings on primary storage and minimize the need to scale your storage on-premises.
- ✓ Tape gateway
    a. Also called as virtual tape library (VTL).
    b. It offers a cost-effective and durable archive backup data in Amazon Glacier.
    c. It provides a virtual tape infrastructure that scales seamlessly with your business needs and

eliminates the operational burden of provisioning, scaling, and maintaining a physical tape infrastructure.

DIRECT CONNECT

- AWS Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS.
- AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard 1-gigabit or 10-gigabit Ethernet fiber-optic cable which provides private connectivity between AWS and your datacenter, office, or colocation environment.
- One end of the cable is connected to your router, the other to an AWS Direct Connect router.

Benefits

- Reduce cost when using large volumes of traffic
- Increase reliability
- Increase bandwidth throughput

VPN VS DIRECT CONNECT

- Vpn connections can be configured in minutes and are a good solution if you have an immediate need, have low to modest bandwidth requirements and can tolerate the inherent variability in internet-based connectivity.
- Direct connect does not involve the internet, instead it uses dedicated, private network connections between your intranet and amazon vpc.

SNOWBALL

- AWS Snowball is a service that accelerates transferring large amounts of data into and out of AWS using physical storage appliances, bypassing the Internet.
- Each AWS Snowball appliance type can transport data at faster-than internet speeds. This transport is done by shipping the data in the appliances through a regional carrier. The appliances are rugged shipping containers, complete with E Ink shipping labels.
- With a Snowball, you can transfer hundreds of terabytes or petabytes of data between your on-premises data centers and Amazon S3.
- AWS Snowball uses Snowball appliances and provides powerful interfaces that you can use to create jobs, transfer data, and track the status of your jobs through to completion.
- Each Snowball is protected by AWS Key Management Service (AWS KMS) and made physically rugged to secure and protect your data while the Snowball is in transit.
- In the US regions, Snowballs come in two sizes: 50 TB and 80 TB. All other regions have 80 TB Snowballs only.
- Once the data is processed and verified, aws preforms a software erasure of the snowball appliance.

  (previously this service was called as import/export service)

Use cases

1. Cloud migration
2. Disaster recovery (from s3 to on premise)
3. Datacenter decommissions
4. Content distribution

Snowball edge

- AWS Snowball Edge is a 100TB data transfer device with on-board storage and compute capabilities.

  (Note: snowball has only storage not compute capabilities, which acts as a aws data center in box. It is also possible to run the lambda function)

- We can use Snowball Edge to move large amounts of data into and out of AWS, as a temporary storage tier for large local datasets, or to support local workloads in remote or offline locations.

- Snowball Edge connects to your existing applications and infrastructure using standard storage interfaces, streamlining the data transfer process and minimizing setup and integration. Snowball Edge can cluster together to form a local storage tier and process your data on-premises, helping ensure your applications continue to run even when they are not able to access the cloud.

Snowmobile

- AWS Snowmobile is an Exabyte-scale data transfer service used to move extremely large amounts of data to AWS. You can transfer up to 100PB per Snowmobile, a 45-foot long ruggedized shipping container, pulled by a semi-trailer truck. Snowmobile makes it easy to move massive volumes of data to the cloud, including video libraries, image repositories, or even a complete data center migration. Transferring data with Snowmobile is secure, fast and cost effective.

- After an initial assessment, a Snowmobile will be transported to your data center and AWS personnel will configure it for you so it can be accessed as a network storage target. When your Snowmobile is on site, AWS personnel will work with your team to connect a removable, high-speed network switch from Snowmobile to your local network and you can begin your high-speed data transfer from any number of sources within your

data center to the Snowmobile. After your data is loaded, Snowmobile is driven back to AWS where your data is imported into Amazon S3 or Amazon Glacier.

- Snowmobile uses multiple layers of security designed to protect your data including dedicated security personnel, GPS tracking, alarm monitoring, 24/7 video surveillance, and an optional escort security vehicle while in transit. All data is encrypted with 256-bit encryption keys managed through the AWS Key Management Service (KMS) and designed to ensure both security and full chain-of-custody of your data.

SQS

- Introduced in 2004 before starting aws.
- Amazon Simple Queue Service (Amazon SQS) is a web service that gives you access to a message queue that can be used to store messages while waiting for a computer to process it.
- SQS offers a reliable, highly-scalable hosted queue for storing messages as they travel between applications or microservices.
- SQS is a distributed queue system that enables web service applications to quickly and reliable queue messages that one component in the application generates to be consumed by another component.
- A queue is a temporary repository for messages that are awaiting processing.
- Messages can contain 256KB of text in any format.
- SQS ensures delivery of each message at least once, and support at least one reader and writer interacting with the same queue.
- A single queue can be used simultaneously by many distributed application component, with no need for those components to coordinate with each other to share the queue.

- Amazon SQS supports both standard and FIFO queues. (A standard queue allows you to have a nearly unlimited number of transactions per second. Standard queues support at-least-once message delivery), Hence it provides an eventual consistency. (without writing it Is not possible to access the content).

  Working

  1. Asynchronously pulls the task messages from the queue.
  2. Retrieves the named file.
  3. Process the conversation.
  4. Writes the image back to s3.
  5. Writes a “task complete” message to another queue.
  6. Deletes the original task message.
  7. Checks for more messages in the work queue.

  SQS REQUEST

- Free for 1st one million requests / month.
- $0.50 per one million thereafter.

  DATA TRANSFER

- Free transfer out for 1st 1GB/month.
- $0.12 per GB/month.

SNS

- Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribe endpoints or clients.
- SNS follow push mechanism to deliver the notification to the client, it eliminates the need of periodically check or poll new information and update.
- SNS can push notifications to mobile devices, email, or any http endpoint.
- To prevent the messages from being lost all messages published to SNS are stored redundantly across multiple AZ.
- SNS allows you to group multiple recipients using topics. A topic is an “access point” for allowing recipients to dynamically subscribe for identical copies of the same notification.
- One topic can support deliveries to multiple endpoints.

  Benefits

  1. Instantaneous, push based delivery
  2. Simple api and easy integration with applications.
  3. flexible message delivery over multiple transport protocol.
  4. Inexpensive, pay-as-you-go model with no up-front costs.
- $ 0.50 per 1 million SNS request.

  ### Practical

  1. Creating a bucket

     Goto s3 -> create a bucket (awstestevent)

2. Creating and attaching a subscription

   Goto sns -> topic -> create topic -> topic name (event) -> other topic action -> edit topic policy -> advanced view -> copy and paste the policy from google (enabling event notifications, make the changes such as region, acc no, topic name, bucket name) -> update policy -> create subscription -> protocol (email) -> endpoint (enter the mail id) -> create subscription.

3. Attaching the event to Bucket

   Goto bucket -> events -> name (event) -> events (any) -> sns topic (select the existing)

(note: Bucket and topic should be in same region)

SES

- Amazon Simple Email Service (Amazon SES) is an email platform that provides an easy, cost-effective way for you to send and receive email using your own email addresses and domains.
- With SES, your aws application can sent as many mail as you need to.
- SES is integrated with aws console so that we can monitor that sending activity.
- We can use Amazon SES to receive mail, you can develop software solutions such as email autoresponders, email unsubscribe systems, and applications that generate customer support tickets from incoming emails.

(note: BOUNCE -> email sender makes a request to SES to send a mail to a recipient if the request is valid it forward the mail to internet and to recipient's ISP if the recipient does not exist ISP sends a

bounce notification to aws.

COMPLAINTS -> The recipients who don’t want to receive the message register a complaint to with ISP, ISP forwards the complaint to aws which is forwarded to sender)

SWF

- The Amazon Simple Workflow Service (Amazon SWF) makes it easy to build applications that use Amazon's cloud to coordinate work across distributed components.
- SWF enables the application for a range of use cases, including media processing web application backends, business process workflows, and analytics pipeline, to be designed as a coordinate of tasks.
- Tasks represents invocations of various processing steps in an application which can be performed by executable code, web service, calls, human actions, and scripts.
- SWF presents a task oriented API, whereas SQS offers a message-oriented API.
- SWF ensures the task is done only once and is never duplicated .
- SWF keeps track of all the tasks and events in an application. With SQS you need to implement your own application-level tracking, especially if your application uses multiple queues.

VPC

- Amazon Virtual Private Cloud (Amazon VPC) enables you to launch Amazon Web Services (AWS) resources into a virtual network that you've defined.
- VPC is a virtual network dedicated to your AWS account. It is logically

isolated from other virtual networks in the AWS cloud

- By VPC, we can have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateway.
- To protect the AWS resources in each subnet, you can use multiple layers of security, including security groups and network access control lists (ACL).

2 types of VPC

1. Default VPC
2. Custom VPC (nondefault VPC)

Default VPC

1. Default VPC is user friendly, allowing you to immediately deploy instances.
2. If you have a default VPC and don't specify a subnet when you launch an instance, the instance is launched into your default VPC.
3. All subnets in default VPC have an internet gateway attached.
4. Each EC2 instance under VPC have private and public IP address.
5. If you delete the default VPC only way to get it back is to contact AWS.

Custom VPC

- VPC which is created by the user according to the custom configuration is called custom VPC.
- Subnets that you create in your nondefault VPC and additional subnets that you create in your default VPC are called *nondefault subnets*.

Subnet

- A *subnet* is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. We can use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that won't be connected to the Internet.

Route table

- A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.
- Each subnet in our VPC must be associated with a route table, the table controls the routing for the subnet.
- A subnet can only be associated with one route table at a time, but we can associate multiple subnets with the same route table.

Internet gateway

- An Internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the Internet.
- An internet gateway provides a route out to the internet.
- An Internet gateway serves two purposes: to provide a target in your VPC route tables for Internet-routable traffic, and to perform network address translation (NAT) for instances that have been assigned public IPv4 addresses.
- For a VPC you can have 1 internet gateway.

NAT

- Network Address Translation (NAT) gateway is used to enable instances in a private subnet to connect to the Internet or other AWS services, but prevent the Internet from initiating a connection with those instances.

NETWORK ACL

- A *network access control list (ACL)* is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets.
- VPC automatically comes with a modifiable default network ACL. By default, it allows all inbound and outbound IPv4 traffic and, if applicable, IPv6 traffic.
- We can create a custom network ACL and associate it with a subnet. By default, each custom network ACL denies all inbound and outbound traffic until you add rules.
- We can associate a network ACL with multiple subnets; however, a subnet can be associated with only one network ACL at a time.
- A network ACL contains a numbered list of rules that we evaluate in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL. The highest number that you can use for a rule is 32766

VPC peering

- A **VPC peering** connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses
- Instances in either **VPC** can communicate with each other as if they are within the same network.
- We can peer VPC with other AWS account as well as with other VPC in the same account, but VPCs must be in the same region.

VPN

- We can connect your VPC to remote networks by using a VPN connection.

- Here we use AWS hardware VPN.

Egress only internet gateway

- An egress-only Internet gateway is a VPC component that allows outbound communication over IPv6 from instances in your VPC to the Internet, and prevents the Internet from initiating an IPv6 connection with your instances.

DHCP Options Set

- The Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network.

VPC endpoint

- A VPC endpoint enables you to create a private connection between your VPC and another AWS service without requiring access over the Internet.
- An endpoint enables instances in your VPC to use their private IP addresses to communicate with resources in other services. Your instances do not require public IPv4 addresses, and you do not need an Internet gateway, a NAT device, or a virtual private gateway in your VPC.
- We use endpoint policies to control access to resources in other services.
- Traffic between your VPC and the AWS service does not leave the Amazon network.

(note: only s3 is supported as of now)

VPC Restrictions

1. 5 elastic IP per VPC.

2. 5 internet gateways per region.
3. 5 VPC per region (can be increased upon request).
4. 5 NAT per region.
5. 5 virtual private gateways per region.
6. 50 VPN connection per region.
7. 50 rules per sg.
8. 50 customer gateways per region. (The customer gateway is the appliance at your end of the VPN connection)
9. 100 security group per VPC.
10. 200 network ACL per region.
11. 200 Route table per region.

PRACTICAL

1. Create VPC

   For creating VPC we have 4 configurations

   1. **VPC with a Single Public Subnet:** The configuration for this scenario includes a virtual private cloud (VPC) with a single public subnet, and an Internet gateway to enable communication over the Internet.
   2. **VPC with Public and Private Subnets:** The configuration for this scenario includes a virtual private cloud (VPC) with a public subnet and a private subnet. (note: A common example is a multi-tier website, with the web servers in a public subnet and the database servers in a private subnet)
   3. **VPC with Public and Private Subnets and Hardware VPN Access:** The configuration for this scenario includes a virtual private cloud (VPC) with a public subnet and a private subnet, and a virtual private

gateway to enable communication with your own network over an IPsec VPN tunnel. (note: This scenario enables you to run a multi-tiered application with a scalable web front end in a public subnet, and to house your data in a private subnet that is connected to your network by an IPsec VPN connection).

(note: **IPSec** is an Internet Engineering Task Force (IETF) standard suite of protocols that provides data authentication, integrity, and confidentiality as data is transferred between communication points across IP networks. **IPSec** provides data security at the IP packet level.)

4. **VPC with a Private Subnet Only and Hardware VPN Access:** The configuration for this scenario includes a virtual private cloud (VPC) with a single private subnet, and a virtual private gateway to enable communication with your own network over an IPsec VPN tunnel.

   (note: we are going to create the subnet from very basics)

   Select your VPC -> create VPC -> give any name -> CIDR (Classless Inter-Domain Routing block is a set of Internet protocol (IP) standards that is used to create unique identifiers for networks and individual devices) (in general case we have 254 host in a n/w but In AWS it is 251, 0 = unicast, 1 = gateway, 2 = dns, 3 = unknown and 255 = broadcast) -> 10.0.0.0/16 -> default tenancy -> create.

2. Create subnet

   Go to the subnet part -> create subnet -> add tag (1a-

public) -> select the custom VPC -> AZ (1a) -> IPV4 CIDR (10.0.1.0/24) -> create subnet -> add tag (1b-public) -> select custom vpc -> AZ (1b) -> IPV4 CIDR (10.0.2.0/24)

3. Spinning an EC2 instance

   (note: Now the subnet that we have created is a private subnet so we won't be able to access internet)

   Create an instance -> attach created VPC -> select subnet (1b) -> Auto assign public ip (even if you assign we won't be able to access) -> default -> add storage -> add tag -> add sg (ssh) -> launch.

4. Creating a route table

   (note: when we create a VPC a route table will be created automatically, by using the existing route table it won't be able to access internet)

   Select the existing route table -> name it as private -> go to subnet -> select (1b-private) -> go to route table option -> edit -> change to custom route table(private) -> save

   Create route table -> name tag(public) -> VPC (select custom VPC) -> go to subnet -> select the custom subnet (1a-public) -> go to route table option -> edit -> change to custom route table(public) -> save

   (note: by using route table we can manage internal traffic)

5. Internet Gateway

   (note: Even if we create an internet gateway it doesn't give you an access to internet, either you need to

elastic load balancer or elastic ip)

Create internet gateway -> name tag(nirmal) -> attach to VPC -> select the VPC -> go to route table (public) -> select routes -> edit -> select target -> select the gateway -> destination 0.0.0.0/0 -> save

6. Elastic IP

   Create an elastic ip

7. Spin a new instance

   Create a new instance -> select the custom VPC -> select the availability zone (1a) -> launch

8. Associate elastic ip

   Go to elastic ip -> associate -> select the instance in public subnet(1a).

9. Connect to the instance

   Login to instance

   (note: here when we check the ip it is the private instance)

10. Login to private instance

    Using winscp copy the public key to public instance -> from public instance use ->

    #chmod 600 <pem file>

#ssh ec2-user@<private ip> -I <pem file>

(note: i -> identity file)

11. Try to install a package

Try to install a package from the instance to verify it is installing or not.

(note: here package won't be installed, now we are going to use, by using NAT it is possible to access network from private instance but not possible to access the instance publicly).

12. Creating NAT gateway

Create NAT -> select the public subnet (1a-public) -> create new elastic ip -> create NAT gateway -> go to subnet -> select public subnet -> go to route table -> edit -> target -> give nat id -> destination -> 0.0.0.0/0

13. Login to instance

Now login to the instance in private subnet -> try to install any package or try to ping to google.

(Note: it should work)

(Note: public subnet -> public route table -> igw, private subnet -> private route table -> nat, we are creating NAT in public instance and attaching to private instance).

14. VPC Peering

Goto peering connection -> create peering connection ->

peering connection name (Checking) -> VPC REQ. (custom vpc) -> VPC ACCEP. (default vpc) -> create peer connection -> goto peering connection menu -> accept the new request -> goto route table of custom vpc (public) -> add the ipv4 cidr and target (172.31.0.0/16 pcx-3e6a9757 connection name) of default vpc -> goto route table of default vpc -> add the ipv4 cidr and target (10.0.0.0/16 connection name) of custom vpc -> save

Now launch an instance in default VPC and try to connect to custom vpc webserver instance (in this example) using private ip, it will be possible, then try to connect to DBS (in this example) it won't be possible to connect, because we have added the entry only in public route table not in private route table.

15. Network ACLs

-

  (note: NetworkACL is primary form of security. It does the same activity of SG, but multiple SG can be under ACL)

  Select the custom acl-> check the subnets which are associated (here it is 2, by default whatever changes we do will be affected on both) -> inbound -> edit -> add another rule -> 95 -> type SSH -> source (give the ip you want to allow) -> allow -> save

  (note: Deny will override allow)

16. Endpoint

    a. Create an IAM user with S3 full access

    b. Launch an amazon instance in custom vpc and private subnet

c. Login and integrate the user by performing

#aws configure (specify the location as ap-south-1)

d. #aws s3 ls (now we will be able to see s3 bucket)

e. Goto private route table and remove nat.

f. Check #aws s3 ls (we won't be able to access)

g. Click on endpoint and associate to private route table

h. Go back to instance and perform #aws s3 ls (now we will be able to see s3 bucket)

RDS

- A **database** is a collection of information that is organized so that it can be easily accessed, managed and updated.
- There are different kinds of database

1. Relational DB

    - A **relational database** is a collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the **database** tables.
    - Amazon have a service called RDS (relational database service) including 6 different db MySQL, MariaDB, Microsoft SQL, Postgres, oracle, Aurora

2. Non-relational DB

    - A non-relational database is any database that does not follow the relational model provided by traditional relational database management systems.
    - DynamoDB is an example

3. Data warehousing DB
    - A data warehouse exists as a layer on top of another database or databases.
    - RedShift is an example.

ElastiCache

- ElastiCache is a web service that makes it easy to set up, manage, and scale a distributed in-memory cache environment in the cloud.
- The service improves the performance of web applications by allowing you to retrieve information from fast, managed, In-memory caches, instead of relying entirely on disk based DB.
- ElastiCache supports two open-source in-memory caching engines
    1. Redis - a fast, open source, in-memory data store and cache.
    2. Memcached - a widely adopted memory object caching system.

Amazon RDS DB Instance

- A *DB instance* is an isolated database environment running in the cloud.
- A DB instance can contain multiple user-created databases.
- We can have up to 40 Amazon RDS DB instances.
- Production environment mainly uses multi AZ deployment, it provides enhanced availability and data durability for instance.
- RDS automatically provision and maintain a synchronous “standby” replica in different AZ.

- RDS automatically fails over to the up-to-date standby database ensuring that database operations resume quickly without administrator intervention, in the event of planned database maintenance or unplanned service disruption.

Read Replica

- It makes it easy for scaling it beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- They can be used for serving read traffic when the primary database is unavailable

DB Snapshot and Automated Backup

- RDS provides 2 ways of backing and restoring your instance
    1. Snapshots
    2. Automated Backup

- Snapshots are user triggered (can be automated via script or application)
- Automated backup are automatic and give the ability to restore point-in-time.
- Both are billable in terms of storage.

RDS Instances Type

- RDS DB instances come in 2 type
    1. Reserved DB instance
    2. On-Demand instance

- 2 instance type are same except billing.
- On-demand is hourly basis.
- Reserved require low up-front, one-time fee and in turn provides a significant discount on the hourly usage charge for the instance.

RDS vs DB on EC2 (check slide)

Practical

1. Launching an instance

We can launch the DB using EC2 OR RDS

1. EC2

Go to ec2 -> launch instance -> rhel -> launch -> login

#yum install -y mariadb*

#yum update -y

#systemctl restart mariadb

#systemctl enable mariadb

2. RDS

Create a security group -> SG name (RDSSecurity) ->Description (RDSSecurity) -> VPC (default) ->add rule -> mysql/aurora (3306) ->add source (copy the SG id of ec2 instance and paste it to source, because we are going to allow only the connection from ec2) -> Go to RDS -> select MariaDB -> dev/test -> DB instance class (t2.micro) -> multi-AZ deployment (no) -> storage type (ssd) -> allocated storage (5G, max 6TB) -> DB instance identifier (nirmal, give a unique name to identify DB) -> master username (nirmal) -> master password (pragathi, give any 8 character) -> confirm password -> VPC (default) -> subnet (default) -> publicly accessible (no, if set to yes the RDS will have a public ip so anyone can access from outside -> VPC SG (RDSSecurity, one that you created) -> DB Name (MariaDB) -> backup retention period (7 days, max 35days, maximum no.of days the snapshot should be retained) -> Backup window (default, specifying at wot time the backup should be done) -> Auto minor version upgrade (yes) -> Launch DB instance.

3.Login to EC2 and connect to RDS

Check mariadb is working fine or not->

#mysql -u root -p

If working

#mysql -h (endpoint, without port no) -P 3306 -u (username) -p

(check whether connection is happening to RDS or not, then show some basic MariaDB cmd)

4. **Restoring DB**

if we restore its going to recreate a new instance with a new endpoint.

Select action -> go to point in time -> use latest restorable time (latest) or custom restore time -> launch.

Amazon Aurora

- Amazon Aurora is a fully managed, MySQL-compatible, relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases
- Amazon Aurora provides 5 times better performance than MySQL, at a price point one tenth of a commercial DB while delivering similar performance and availability.
- Amazon Aurora default size is 10GB max it can scale upto 64TB.
- Compute resource can scale upto 32 vCPUs and 244 GB of memory.

DynamoDB

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- We can use Amazon DynamoDB to create a database table that can store and retrieve any amount of data, and serve any level of request traffic.

- Amazon DynamoDB automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent and fast performance.
- Its flexible data model and reliable performance make it a great fit for the mobile, web, gaming and many other applications.
- It always stores on SSD storage there is no magnetic storage.
- Spreads data to multiple AZ.

PRACTICAL

1. Create table

   Go to DynamoDB -> create table -> table name (Music) -> primary key (Artist) -> add sort key (Song title) -> select the table name -> goto item -> create item -> add artist string (Eminem) -> song title (not afraid) -> add new items by clicking on + symbol -> append -> string (Album title) -> string (recovery) -> add new item -> year (2010) -> save (**similarly create 2 more**).

   (note: **string** is any finite sequence of characters)

2. Edit table

   Goto item -> action -> edit -> give the appropriate value

3. Query the data

   Select the table -> item -> click the drop down labeled scan -> on the drop down change the scan to query -> in artist enter the name of artist -> and scan.

Redshift

- Amazon Redshift is a fast and powerful, fully managed, petabyte-scale data warehouse service in the cloud.
- It provides a simple and cost-effective way to analyze all your data using existing Business Intelligence (BI) tool and SQL clients, regardless of the size of data.
- It is designed to handle datasets from few hundred giga byte to a petabyte or more
- Customers can start small for just $0.25 per hour with no commitments or upfront cost and scale to a petabyte or more for $1000 or more tera byte per year, less than a 10^{th} of most other data base solution.
- Amazon Redshift handles all the data warehouse management activities for you, from provisioning the infrastructure to automating ongoing administrative tasks such as backup and patching.
- Amazon Redshift is 10 times faster than traditional warehousing solution, because it stores the data in columnar form (columnar data storage). i.e. instead of storing data as a series of rows, amazon redshift organize data by column.
- Data transferred in redshift is encrypted with SSL, data stored is encrypted by AES 256.

(note: we use BI or some web interface to use redshift)

Route 53

- DNS is a hierarchical distributed naming system for machines connected to a network, it enables to map a human readable name to a machines ip address.
- Route 53 is DNS service of AWS.
- Amazon Route 53 is highly available and scalable cloud domain name system (DNS) web service, named after port 53 which is the DNS port.
- It provides secure routing connection to aws service such as EC2, ELB, S3.
- Route 53 is not limited to AWS infrastructure you can manage our DNS record through Route 53.
- Route 53 is global service.

Benefits

1. Fast, Reliable and cost effective since it uses edge location.
2. It is 100% available.
3. It’s a pay per use.

Practical

1. Create a WordPress

Create an ec2 instance with WordPress -> attach an elastic ip-> login to website -> add a file from s3 -> check whether the website is working or not.

2. **Login to freenom**

   Goto freenom -> check whether the name is available or not -> check the available name -> select -> continue -> login using google, facebook or live > compete the form.

3. **Route 53**

   Go to Route53 -> DNS management -> create hosted zone -> domain name (nirmal.ga, give the domain name from freenom that you have created) -> type (public hosted zone) -> create.

   - A **hosted zone** is a collection of resource record sets for a specified domain.
   - A **resource record** is an entry in DNS zone that specifies information about a particular name or object in the zone.

4. **Go to freenom**

   Go to freenom -> service -> my domain -> manage domain -> management tools -> name server -> use custom name server -> copy the values from hosted zone -> paste it to name server -> change name server.

   (note: it may take a while to for the domain name provider to map the AWS name server with the domain name.)

5. **Configuring domain Record Set**

   Select the hosted zone -> create record set -> name (www) -> value (elastic ip or ip of the instance) -> create.

- **Resource set** tell the DNS how to how you want traffic to be routed for that domain.

  (note: here if you click on alias -> yes, we can see alias target there we can specify S3, ELB, CloudFront etc.)

- **Routing Policy** determines how Amazon Route 53 responds to queries.

  1. **Simple Routing Policy:** it is used when we have a single resource that performs a given function for your domain

  2. **Weighted Routing Policy:** it is used when you have multiple resources that perform the same function.

  3. **Latency Routing Policy:** it is used when we have resources in multiple Amazon EC2 data centers that perform the same function and you want Amazon Route 53 to respond to DNS queries with the resources that provide the best latency.

  4. **Failover Routing Policy:** it is used when we want to configure active-passive failover, in which one resource takes all traffic when it's available and the other resource takes all traffic when the first resource isn't available.

  5. **Geolocation Routing Policy:** it is used when we want Amazon Route 53 to respond to DNS queries based on the location of your users.

6. **Verify the web page**

   Open a tab -> www.nirmal.ga.

7. Health Check

- Route 53 health checks monitor the health and performance of your application's servers, or endpoints, from a network of health checkers in locations around the world.
- we can specify either a domain name or an IP address and a port to create HTTP, HTTPS, and TCP health checks that check the health of the endpoint.
- we can use Route 53 health checks for monitoring and alerts. Each health check provides CloudWatch metrics that you can view and set alarms on.
- We can also use Route 53 health checks for DNS failover by associating health checks with any Route 53 DNS resource record set. This lets you route requests based on the health of your endpoints.

  (note: here we are going to **create health check with routing policy failover**).

1. Create instance

Create 2 instance with httpd web server -> one in Mumbai (Health Check 1, content) and other in different region (Singapore, Health Check 2, content).

2. Create health check

Go to health check -> create health check -> name (regionalhealthcheck) ->specific endpoint (ip) -> protocol (http) -> ip addrs (ip of the 1st instance, mumbai) -> hostname (webprimary)

-> port (80) -> path (index.html) -> next -> cloudwatch (yes) -> create a topic -> create.

3. Create a primary record set

Go to hosted zone -> create a record set -> name (www2) -> TTL (1m) -> value (ip of 1st instance) -> routing policy (Failover) -> Failover record type (primary) -> set ID (www2 -primary, created by default) -> Associate with health check (yes) -> Health check to associate (regionalhealthcheck, one which we create previously) -> save record set.

(note: after 5 minute's check www2.nirmal.ga, it will display "Health Check 1").

Go to hosted zone -> create a record set -> name (www2) -> TTL (1m) -> value (ip of 2nd instance) -> routing policy (Failover) -> Failover record type (secondary) -> set ID (www2-Secondary, created by default) -> Associate with health check (no) -> save record set.

(note: stop the 1st instance after 5 minute's checks www2.nirmal.ga, it will display "Health Check 2").

CloudFormation

- AWS CloudFormation allows you to quickly and easily deploy your infrastructure resources and applications on AWS.
- It simplifies provisioning and managing resources on aws.

- We can create template for the services and applications you want to build on aws.
- AWS CloudFormation uses those templates to quickly and reliably provision those services or applications, called stacks.
- We can use resources from over 20 aws services such as EC2, VPC, RDS, Redshift etc in CloudFormation.

Practical

Goto cloudformation -> create stack -> select a sample stack (single instance sample – wordpress blog) -> stack name (sample) -> DBPassword (pragathi) -> DBRootPassword (pragathi) -> DBUser (nirmal) -> instance type (t2. micro) -> keyname (select the existing key) -> create.

(Here we are going to represent how to create a webserver instance by simple stack)

Amazon Elastic Transcoder

- Amazon Elastic Transcoder lets you convert media files that you have stored in S3 into media files in the formats required by consumer playback devices.
- Pay based on the minutes that we transcode and the resolution at which we transcode.

  (note: For example, you can convert large, high-quality digital media files into formats that users can play back on mobile devices, tablets, web browsers, and connected televisions.)

- Elastic Transcoder has four components:
  1. **Jobs**: Each job converts one file into up to 30 formats.

(For example, if you want to convert a media file into six different formats, you can create files in all six formats by creating a single job.
When you create a job, you specify the name of the file that you want to transcode, the names that you want Elastic Transcoder to give to the transcoded files, and several other settings)

2. **Pipelines** are queues that manage your transcoding jobs. A pipeline can process more than one job simultaneously. We can temporarily stop processing jobs by pausing it
(When you create a job, you specify which pipeline you want to add the job to. Elastic Transcoder starts processing the jobs in a pipeline in the order in which you added them. If you configure a job to transcode into more than one format, Elastic Transcoder creates the files for each format in the order in which you specify the formats in the job.)
3. **Presets** are templates that contain most of the settings for transcoding media files from one format to another.
(Elastic Transcoder includes some default presets for common formats, for example, several iPod and iPhone versions.)
4. **Notifications** let you optionally configure Elastic Transcoder and Amazon SNS to keep you apprised of the status of a job **(**when Elastic Transcoder starts processing the job, when Elastic Transcoder finishes the job, and whether Elastic Transcoder encounters warning or error conditions during processing. Notifications eliminate the need for polling to determine when a job has finished. You configure notifications when you create a pipeline.)

Practical

Create 2 bucket one for source and other destination -> add a video to source bucket -> goto elastic transcoder console -> create a new pipeline -> pipeline name -> input bucket -> iam role (by default a role will be attached) -> destination

bucket -> class (user wish) -> create pipeline
Create a job -> select the existing pipeline -> input key (name of the file) -> preset (define the video resolution) -> output key (name the output file, but at the end we should give .mp4 extension) -> create job
To check whether job is completed or not click on jobs -> select the pipeline -> click on search.
After that goto destination bucket and check the video is transcoded or not.

RESOURCE GROUPS

- Resource Groups helps to create a custom console that organizes and consolidates information based on your project and the resources that we use.
- If we manage resources in multiple regions, we can create a resource group to view resources from different regions on the same page.
- Resource Groups can display metrics, alarms, and configuration details.

Practical

Create 2 instance in different region -> create a tag with same key and different value -> resource group -> group name (sample) -> tags (select the key and corresponding values) -> resource type -> save.

Lambda

- AWS Lambda is a compute service that runs your code in response to events and automatically manages the underlying compute resource for you.
- It can automatically run code in response to modifications to objects in S3 bucket, messages arriving in kinesis stream, or table update in dynamo DB.
- AWS Lambda lets you run code without provisioning or managing servers.
- You pay only for the compute time you consume - there is no charge when your code is not running.
- Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.
- You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.
- In short lambda is a service to run your code, all you need is to supply the code.
- Supported programming language is JavaScript.
- It is designed to provide 99.99% availability.

Pricing

- First 1st million requests is free there after $0.20.
- Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to nearest 100ms.
- The price depends upon the amount of memory you allocate to your function. You are charged $0.00001667 for every GB

used.

Practical

1. Run a serverless “Hello World”

    Goto lambda -> in filter type “hello-world-python” -> create -> name (You can name your lambda function here. For this tutorial, enter hello-world-python) -> Description (You can enter a short description of your function here. This is pre-populated with A starter AWS Lambda Function.) -> Runtime (Currently, you can author your Lambda function code in Java, Node.js, or Python 2.7. For this tutorial, leave this on *Python 2.7* as the runtime.) -> Lambda function code (you can review the example code authored in Python.) -> Handler (is a method/function in your code, where AWS Lambda can begin executing your code.) -> Role (