**Lecture Notes for AWS Developer**

**Module-2**

**Amazon Route 53**

Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost effective way to route end users to Internet applications by translating names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other. Amazon Route 53 is fully compliant with IPv6 as well.

* Amazon Route 53 effectively connects user requests to infrastructure running in AWS – such as Amazon EC2 instances, Elastic Load Balancing load balancers, or Amazon S3 buckets – and can also be used to route users to infrastructure outside of AWS.
* You can use Amazon Route 53 to configure DNS health checks to route traffic to healthy endpoints or to independently monitor the health of your application and its endpoints.
* Amazon Route 53 Traffic Flow makes it easy for you to manage traffic globally through a variety of routing types, including Latency Based Routing, Geo DNS, Geoproximity, and Weighted Round Robin—all of which can be combined with DNS Failover in order to enable a variety of low-latency, fault-tolerant architectures.
* Using Amazon Route 53 Traffic Flow’s simple visual editor, you can easily manage how your end-users are routed to your application’s endpoints—whether in a single AWS region or distributed around the globe.
* Amazon Route 53 also offers Domain Name Registration – you can purchase and manage domain names such as example.com and Amazon Route 53 will automatically configure DNS settings for your domains.

**Benefits**

**Highly available and reliable**

Amazon Route 53 is built using AWS’s highly available and reliable infrastructure. The distributed nature of our DNS servers helps ensure a consistent ability to route your end users to your application.

**Flexible**

Amazon Route 53 Traffic Flow routes traffic based on multiple criteria, such as endpoint health, geographic location, and latency. You can configure multiple traffic policies and decide which policies are active at any given time. You can create and edit traffic policies using the simple visual editor in the Route 53 console, AWS SDKs, or the Route 53 API.

**Designed for use with other Amazon Web Services**

Amazon Route 53 is designed to work well with other AWS features and offerings. You can use Amazon Route 53 to map domain names to your Amazon EC2 instances, Amazon S3 buckets, Amazon CloudFront distributions, and other AWS resources.

**Simple**

With self-service sign-up, Amazon Route 53 can start to answer your DNS queries within minutes. You can configure your DNS settings with the AWS Management Console or our easy-to-use API. You can also programmatically integrate the Amazon Route 53 API into your overall web application.

**Fast**

Amazon Route 53 Traffic Flow lets you further improve your customers’ experience by running your application in multiple locations around the world and using traffic policies to ensure your end users are routed to the closest healthy endpoint for your application.

**Cost-effective**

Amazon Route 53 passes on the benefits of AWS’s scale to you.  You pay only for the resources you use, such as the number of queries that the service answers for each of your domains, hosted zones for managing domains through the service, and optional features such as traffic policies and health checks, all at a low cost and without minimum usage commitments or any up-front fees.

**Secure**

By integrating Amazon Route 53 with AWS Identity and Access Management (IAM), you can grant unique credentials and manage permissions for every user within your AWS account and specify who has access to which parts of the Amazon Route 53 service.

**Scalable**

Route 53 is designed to automatically scale to handle very large query volumes without any intervention from you.

**Simplify the hybrid cloud**

Amazon Route 53 Resolver provides recursive DNS for your Amazon VPC and on-premises networks over AWS Direct Connect or AWS Managed VPN.

**Amazon Virtual Private Cloud (Amazon VPC)**

Amazon Virtual Private Cloud (Amazon VPC) is a service that lets you launch AWS resources in a logically isolated virtual network that you define. You 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 gateways. You can use both IPv4 and IPv6 for most resources in your virtual private cloud, helping to ensure secure and easy access to resources and applications.

**Benefits of Using Amazon Virtual Private Cloud (Amazon VPC)**

* Secure and monitored network connections
* Simple set-up and use
* Customizable virtual network

**Use cases**

* Host a simple, public-facing website
* Host multi-tier web applications
* Back up and recover your data after a disaster
* Extend your corporate network into the cloud
* Securely connect cloud applications to your datacenter

**VPC peering**

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



AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck.

A VPC peering connection helps you to facilitate the transfer of data. For example, if you have more than one AWS account, you can peer the VPCs across those accounts to create a file sharing network. You can also use a VPC peering connection to allow other VPCs to access resources you have in one of your VPCs.

**VPN Vs VPC**

* A **VPN** makes the private network (such as a company network) of an entity accessible through public infrastructure, primarily the internet. A VPN can allow users to exchange data efficiently across shared or public networks, as though they are directly linked to the private network.
* Commercial VPNs secure traffic in an encrypted tunnel, then route that traffic through a server in a remote location of the user’s choosing. This masks the location, IP address, and online activity of the user.
* A VPN (Virtual Private Network) is essentially an encrypted “channel” connecting two networks, or a machine to a network, generally over the public internet.
* A **VPC** (Virtual Private Cloud) is a virtual network in AWS (Amazon Web Services). It can be divided into private and public subnets have custom routing rules, have internal connections to other VPCs, etc. EC2 instances and other resources are placed in VPCs similarly to how physical data centers have operated for a very long time.
* A VPC can manage large numbers of virtual machines and are not limited to a single, fixed-resource server. Users are not bound by the limitation sof the underlying hardware.
* VPCs allow their users to manage their own service.

**STEPS TO CREATE VPC USING AWS CLI**

**STEP1:**

For general use, the **aws configure** command is the fastest way to set up your AWS CLI installation. When you enter this command, the AWS CLI prompts you for four pieces of information:

* Access key ID
* Secret access key
* AWS Region
* Output format

Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS.

The Default region name identifies the AWS Region whose servers you want to send your requests to by default.

The Default output format specifies how the results are formatted. The value can be any of the values in the following list. If you don't specify an output format, json is used as the default.

**FOR EXAMPLE**

$ **aws configure**

AWS Access Key ID [None]: *AKIAIOSFODNN7EXAMPLE*

AWS Secret Access Key [None]: *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

Default region name [None]: *ap-south-1*

Default output format [None]: *json*

**STEP-2**

**To create VPC using CIDR block 10.0.0.0/16 the command is:**

aws ec2 create-vpc --cidr-block 10.0.0.0/16

**In the output that's returned, take note of the VPC ID.**

{

 "Vpc": {

 "VpcId": "vpc-2f09a348",

 ...

 }

}

**AWS CLI for Security Group and S3**

**To create SG for VPC**

aws ec2 create-security-group --group-name my-sg11 --description "My security group1" --vpc-id vpc-0d0337664aaa0ab93

**To describe**

aws ec2 describe-security-groups --group-ids sg-0711db3fd7d5f5173

**To create SG for EC2**

aws ec2 create-security-group --group-name my-sg12 --description "My security group12"

**To describe different VPCs**

aws describe -vpcs - -vpc-ids

**For AWS Documentation AWS CLI for S3**

https://docs.aws.amazon.com/cli/latest/userguide/cli-services-s3-commands.html

**Copy an object from one bucket to another**

aws s3 cp s3://sfsg1/index1.jpeg s3://mybucket148/

**Copy an object to a bucket**

cd Desktop

dir

aws s3 cp page1.docx s3://mybucket148

**Move an object from one bucket to another**

aws s3 mv s3://sfsg1/index2.jpeg s3://mybucket148/

**Move an object to a bucket**

cd Desktop

dir

aws s3 mv page2.docx s3://mybucket148