**Route53**

AWS DNS Service (Domain Name System)

Internet traffic – Public internet traffic - Example (IGW)

Internal VPC - VPN only access eg. (NAT)

* Domain Name registration ([www.example.com](http://www.example.com))
* Route internet traffic
* Health Check for Web Resources

1. Domain Name registration
   1. [www.example.com](http://www.example.com), [www.example.org](http://www.example.org) ..........
2. Route internet traffic
   1. Hosted Zone
      1. Multiple records
         1. A Record 🡪 IPv4 IP
         2. AAAA Record 🡪 IPv6 IP Address
         3. NS 🡪 Name Servers
         4. SOA 🡪 Start of Authority
         5. MX – Mail exchange
         6. CName 🡪 Canonical Name
3. Health Check for Resources
   1. 80 🡪 http
   2. 443 🡪 https

Routing Policies in Route53

1. Simple Policy

**Simple Routing Policy:**

Use a simple routing policy when you have a single resource that performs a

given function for your domain, for example, one web server that serves

content for the example.com website. In this case, Amazon Route 53 responds

to DNS queries based only on the values in the resource record set, for example, the IP address in an A record.

1. Weighted Routing Policy

**Weighted Routing Policy:**

Use the weighted routing policy when you have multiple resources that

perform the same function (for example, web servers that serve the same

website) and you want Amazon Route 53 to route traffic to those resources in

proportions that you specify (for example, one quarter to one server and three quarters to the other).

1. Low latency Routing Policy 🡪 Less query and faster response time

**Latency Routing Policy:**

Use the latency routing policy when you 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.

For example, you might have web servers for example.com in the Amazon EC2

data centers in Ireland and in Tokyo. When a user browses to example.com,

Amazon Route 53 chooses to respond to the DNS query based on which data center gives your user the lowest latency.

1. Failover Routing Policy

**Failover Routing Policy (Public Hosted Zones Only):**

Use the failover routing policy when you 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.

1. Geolocation Routing Policy

**Geolocation Routing Policy:**

Use the geolocation routing policy when you want Amazon Route 53 to respond to DNS queries based on the location of your users.

**LAB Route53:**

* + - 1. Create web application – WordPress website – Hello World
      2. Make sure SG is configured for web access
      3. Create a Load balancer
      4. Create a Target Group
      5. Create a hosted zone in Route53
      6. Configure out domain to host our web application
         1. Public IP Address of our web server.
         2. DNS of our Load balancer.

**Storage on the cloud**

1. S3 (Simple storage service) 🡪 Object or file storage 🡪 Cost effective, high availability, durability.
   1. S3 Bucket
   2. Host a static website
   3. Backup and recovery
   4. Store log files
   5. Migrate applications file and vm files from one environment to another or from on prem to cloud
   6. Host **hybrid** or **on-prem** storage need (Have applications running on-prem and utilizing AWS S3 as their storage using storage gateway)
   7. Store already encrypted data and encrypt
   8. Use as a **versioning** storage service
      * 1. Having multiple versions of the same object

LAB S3 Static Website:

1. Create an S3 bucket and host a static website using index.html

**Questions.**

1. How can you delete a bucket in a real life situation when the bucket is already in used. Instead of updating? By creating bucket policy
2. Is it possible to automate older versions of files within a bucket to store in glacier or a cheap storage? 🡪 Lambda Functions

**LAB (Host Static Website in S3 bucket):**

1. Create an S3 bucket and host a static website using index.html
   * + - 1. Create an S3 bucket
         2. Upload our html file into S3 bucket
         3. Edit permission to allow access public to bucket
         4. Configured bucket to host static website in properties
         5. Provide bucket policy if necessary
         6. Make our html file public
2. EBS (Elastic Block storage) 🡪 Block Storage
3. EFS (Elastic File Storage) 🡪 File storage

**LAB (EFS):**

Create EFS and mount same EFS file on 2 private instances.

VPC NID 🡪 192.168.1.0/16

1. 1 Public Subnet in AZ1a NID 🡪 192.168.1.0/26 🡪 Bastion Host (Jumpbox)
2. 1 Private Subnet in AZ1b NID 🡪 192.168.1.64.0/26 in private network
3. 1 Private subnet in AZ1c NID 🡪 192.168.1.128.0/16 in private network

By default VPC will

1. RT1 🡪 Public
2. SG
3. Network ACL (Do not need)

Things we will create

1. Subnet (Total 3 subnets)
2. Private RT
3. IGW
4. NAT
5. Configure our RT

Create EFS.

* To mount to our private C2 instances.

Create EC2 instances

EC2 in a public subnet AZ1a

1 private EC2 in private subnet AZ1b

1 Private EC2 in private subnet AZ1c

Ec2 instances need to have same SG as efs-vpc

Mount EFS unto instance – Instructions