VPC, CNAME TYPE ROUTING, ELB, PEERING, FAILOVER ROUTING POLICY ASSIGNMENT

-SUBMITTED BY: KALAISELVAN P

1. **Create vpc with one bastion host and webserver in public subnet and one or two appservers in private subnet. Connect from bastionhost to webserver and install httpd with a web page. From ther again connect to appserver1 and appserver2 to install httpd server and display the web content with load balancer and route 53.**

   Step 1: create VPC with a public and a private subnet.

   Step2: create two servers in public subnet (**1. Bastionhost, 2. Webserver**) and create two appservers in the private subnet (**appserver1 and appserver2**)

   Step 3: login to **bastion host** and connect to the webserver in the public subnet

   Step 4: Installed httpd in webserver and hosted the content "**Welcome To The Webserver at Public Subnet**"

   Step 5: connected to the appserver 1(natgateway enabled) in the private subnet**, installed httpd** and hosted the content "**welcome to the appserver 1 through httpd**"

   Step 6: connected to the appserver2 (natgateway enabled) in the private subnet, **installed nginx** and hosted the content "**welcome to the nginx server**"

   Step 7: Connected the instances to the load balancer and obtained the dns obtained from the load balancer. Tested content in the browser.

   Step 8: route53 (dns) is used to create hosted zone and records with **cname type routing** and mapped the nameservers in the godaddy domain page. Domain name was hit in the browser to check the content of all the servers.

   Step3 output

   login to bastion host and connect to the webserver in the public subnet

Connected from the bastion host to the webserver in the public subnet

```
[root@ip-10-0-1-137 ~]# chmod 400 test.pem
[root@ip-10-0-1-137 ~]# ssh -i test.pem ec2-user@10.0.1.92

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-1-92 ~]$
```

Step 4 output

Install httpd in webserver and hosted the content “Welcome To The Webserver at Public Subnet”

Httpd installed in the webserver at public subnet. Public ip gives connectivity to webserver in public subnet

```
   Active: active (running) since Tue 2023-07-11 10:10:41 UTC; 14s ago
     Docs: man:httpd.service(8)
 Main PID: 3625 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes se
rved/sec:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─3625 /usr/sbin/httpd -DFOREGROUND
           ├─3626 /usr/sbin/httpd -DFOREGROUND
           ├─3627 /usr/sbin/httpd -DFOREGROUND
           ├─3628 /usr/sbin/httpd -DFOREGROUND
           ├─3629 /usr/sbin/httpd -DFOREGROUND
           └─3630 /usr/sbin/httpd -DFOREGROUND

Jul 11 10:10:41 ip-10-0-1-92.ap-south-1.compute.internal systemd[1]: Starting...
Jul 11 10:10:41 ip-10-0-1-92.ap-south-1.compute.internal systemd[1]: Started ...
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-10-0-1-92 ~]# cd /var/www/html/
[root@ip-10-0-1-92 html]# vi index.html
[root@ip-10-0-1-92 html]#
```

Static content is hosted :"Welcome To The Webserver at Public Subnet"

Step 5 output :

Connected to the appserver 1(natgateway enabled) in the private subnet, installed httpd and hosted the content “welcome to the appserver 1 through httpd”

With natgateway connected to appserver 1 and appserver 2 and installed httpd in appserver1 and tomcat in appserver2

```
[ec2-user@ip-10-0-1-92 ~]$ sud0 -i
-bash: sud0: command not found
[ec2-user@ip-10-0-1-92 ~]$ sudo -i
[root@ip-10-0-1-92 ~]# vi test.pem
[root@ip-10-0-1-92 ~]# chmod 400 test.pem
[root@ip-10-0-1-92 ~]#  ssh -i test.pem ec2-user@10.0.3.180
The authenticity of host '10.0.3.180 (10.0.3.180)' can't be established.
ECDSA key fingerprint is SHA256:JDfaoaaQV4kM2ucCVgI2UU/1mowlBrRZU0Ipd6h3acY.
ECDSA key fingerprint is MD5:3a:0c:15:bf:83:ab:c2:df:9d:75:18:48:2c:f8:7b:a3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.3.180' (ECDSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-3-180 ~]$ sudo -i
[root@ip-10-0-3-180 ~]# yum install httpd
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
amzn2extra-docker                                        | 3.0 kB     00:00
amzn2extra-kernel-5.10                                   | 3.0 kB     00:00
```

Step 6 output:

connected to the appserver2 (natgateway enabled) in the private subnet, installed nginx and hosted the content “welcome to the nginx server”

Connected from appserver1 to appserver 2

```
[ec2-user@ip-10-0-3-180 ~]$ sudo -i
[root@ip-10-0-3-180 ~]# vi test.pem
[root@ip-10-0-3-180 ~]# chmod 400 test.pem
[root@ip-10-0-3-180 ~]# ssh -i test.pem ec2-user@10.0.3.147
The authenticity of host '10.0.3.147 (10.0.3.147)' can't be established.
ECDSA key fingerprint is SHA256:TJCgLMdUd/y0D4IROU0s4lHN75KOyBVkB1M/pRsmXEA.
ECDSA key fingerprint is MD5:6f:11:24:7a:c0:ca:17:f5:a2:ce:10:ca:14:22:77:05.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.3.147' (ECDSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-3-147 ~]$
```

Nginx installed and started in appserver 2

```
[root@ip-10-0-3-147 bin]# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-9.0.78
Using CATALINA_HOME:   /opt/apache-tomcat-9.0.78
Using CATALINA_TMPDIR: /opt/apache-tomcat-9.0.78/temp
Using JRE_HOME:        /
Using CLASSPATH:       /opt/apache-tomcat-9.0.78/bin/bootstrap.jar:/opt/apache-tomcat-9.0.78/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

Step 7 output:

Connected the instances to the load balancer and obtained the dns obtained from the load balancer. Tested content in the browser.

Step 8 output:

Route53 (dns) is used to create hosted zone and records with cname type routing and mapped the nameservers in the godaddy domain page. Domain name was hit in the browser to check the content of all the servers.

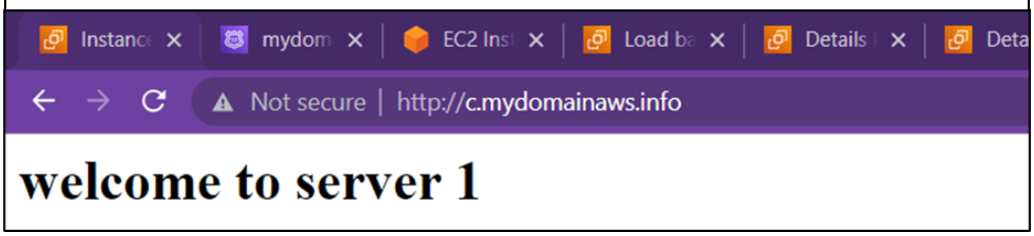
ns-1731.awsdns-24.co.uk

ns-262.awsdns-32.com

ns-1519.awsdns-61.org

ns-1013.awsdns-62.net

Using cname type routing, using DNS of ELB



2. **VPC peering**

   Same region:

   Step1: created two VPCs in same region (10.0.0.0/16 and 180.0.0.0/16)

   Step2: request for peering has been given from VPC-1

   Step3: At VPC 2, accepted the request obtained from the VPC-1

   Step 4: At routetable-1 (vpc1), added a route for cidr:180.0.0.0/16 with peering condition and at routetable-2(vpc2), added a route for cidr:10.0.0.0/16 with peering condition.

   Step 5: For instance 1(vpc1), ssh and all traffic connection (180.0.0.0/16) were rules added in security group

   Step 6: For instance 2 (vpc2), ssh and all traffic connection (10.0.0.0/16) were rules added in security group

   Using cli: ping 180.0.0.0/16 and ping 10.0.0.0/16 were given from their respective instances for the connection. And output came successful.

```
[root@ip-10-0-0-17 ~]# ping 180.0.1.63
PING 180.0.1.63 (180.0.1.63) 56(84) bytes of data.
64 bytes from 180.0.1.63: icmp_seq=43 ttl=255 time=0.742 ms
64 bytes from 180.0.1.63: icmp_seq=44 ttl=255 time=0.448 ms
64 bytes from 180.0.1.63: icmp_seq=45 ttl=255 time=0.419 ms
64 bytes from 180.0.1.63: icmp_seq=46 ttl=255 time=0.415 ms
64 bytes from 180.0.1.63: icmp_seq=47 ttl=255 time=0.399 ms
64 bytes from 180.0.1.63: icmp_seq=48 ttl=255 time=0.429 ms
64 bytes from 180.0.1.63: icmp_seq=49 ttl=255 time=0.394 ms
64 bytes from 180.0.1.63: icmp_seq=50 ttl=255 time=0.462 ms
64 bytes from 180.0.1.63: icmp_seq=51 ttl=255 time=0.467 ms
64 bytes from 180.0.1.63: icmp_seq=52 ttl=255 time=0.398 ms
64 bytes from 180.0.1.63: icmp_seq=53 ttl=255 time=0.430 ms
64 bytes from 180.0.1.63: icmp_seq=54 ttl=255 time=0.475 ms
64 bytes from 180.0.1.63: icmp_seq=55 ttl=255 time=0.423 ms
64 bytes from 180.0.1.63: icmp_seq=56 ttl=255 time=0.409 ms
^C
--- 180.0.1.63 ping statistics ---
56 packets transmitted, 14 received, 75% packet loss, time 56307ms
rtt min/avg/max/mdev = 0.394/0.450/0.742/0.088 ms
[root@ip-10-0-0-17 ~]#
```

```
[root@ip-180-0-1-63 ~]# ping 10.0.0.17
PING 10.0.0.17 (10.0.0.17) 56(84) bytes of data.
^C
--- 10.0.0.17 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3067ms

[root@ip-180-0-1-63 ~]# ping 10.0.0.17
PING 10.0.0.17 (10.0.0.17) 56(84) bytes of data.
64 bytes from 10.0.0.17: icmp_seq=85 ttl=255 time=0.447 ms
64 bytes from 10.0.0.17: icmp_seq=86 ttl=255 time=0.543 ms
64 bytes from 10.0.0.17: icmp_seq=87 ttl=255 time=0.452 ms
64 bytes from 10.0.0.17: icmp_seq=88 ttl=255 time=0.435 ms
64 bytes from 10.0.0.17: icmp_seq=89 ttl=255 time=0.423 ms
64 bytes from 10.0.0.17: icmp_seq=90 ttl=255 time=0.444 ms
64 bytes from 10.0.0.17: icmp_seq=91 ttl=255 time=0.393 ms
64 bytes from 10.0.0.17: icmp_seq=92 ttl=255 time=0.406 ms
64 bytes from 10.0.0.17: icmp_seq=93 ttl=255 time=0.477 ms
^C
--- 10.0.0.17 ping statistics ---
93 packets transmitted, 9 received, 90% packet loss, time 94191ms
rtt min/avg/max/mdev = 0.393/0.446/0.543/0.048 ms
[root@ip-180-0-1-63 ~]#
```

Peering between different regions:

Step 1: created two VPCs in two different regions (10.0.0.0/16 (mumbai) and 180.0.0.0/16 (singapore))

Step 2: request for peering has been given from VPC-1

Step 3: At VPC 2, accepted the request obtained from the VPC-1

Step 4: At routetable-1 (vpc1), added a route for cidr:180.0.0.0/16 with peering condition and at routetable-2(vpc2), added a route for cidr:10.0.0.0/16 with peering condition.

Step 5: For instance 1(vpc1), ssh and all traffic connection (180.0.0.0/16) were rules added in security group

Step 6: For instance 2 (vpc2), ssh and all traffic connection (10.0.0.0/16) were rules added in security group

Using cli: ping 180.0.0.0/16 and ping 10.0.0.0/16 were given from their respective instances for the connection. And output came successful.

```
[root@ip-10-0-13-213 ~]# ping 180.0.1.63
PING 180.0.1.63 (180.0.1.63) 56(84) bytes of data.
64 bytes from 180.0.1.63: icmp_seq=1 ttl=255 time=65.6 ms
64 bytes from 180.0.1.63: icmp_seq=2 ttl=255 time=65.7 ms
64 bytes from 180.0.1.63: icmp_seq=3 ttl=255 time=65.6 ms
64 bytes from 180.0.1.63: icmp_seq=4 ttl=255 time=65.7 ms
64 bytes from 180.0.1.63: icmp_seq=5 ttl=255 time=65.6 ms
64 bytes from 180.0.1.63: icmp_seq=6 ttl=255 time=65.7 ms
64 bytes from 180.0.1.63: icmp_seq=7 ttl=255 time=65.6 ms
64 bytes from 180.0.1.63: icmp_seq=8 ttl=255 time=67.5 ms
64 bytes from 180.0.1.63: icmp_seq=9 ttl=255 time=65.7 ms
64 bytes from 180.0.1.63: icmp_seq=10 ttl=255 time=65.6 ms
^C
--- 180.0.1.63 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 65.616/65.855/67.582/0.588 ms
[root@ip-10-0-13-213 ~]#
```

==========

3.Failover Routing Policy

Step1: Two servers created. Httpd installed in primary server with the static content "welcome to primary data center". Ingnx installed in another server with static content "welcome to disaster recovery".

Step 2: Route53 (dns) is used to create hosted zone and mapped the nameservers in the godaddy domain page.

Step 3: 1st Record with A type and failover routing policy, kept the primary server ip as primary type . created health checks and mapped in the records.

Step 4: 2nd Record with A type and failover routing policy, kept the secondary server ip as secondary type and created.

Step 5: Domain name was hit in the browser to check the content of the primary server(datacenter).

Step 6: The primary server was intentionally made down to check the routing of traffics to secondary server (disaster recovery)

Name servers obtained from the hosted zone and mapped with go daddy domain

ns-830.awsdns-39.net

ns-2006.awsdns-58.co.uk

ns-114.awsdns-14.com

ns-1107.awsdns-10.org