**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**”

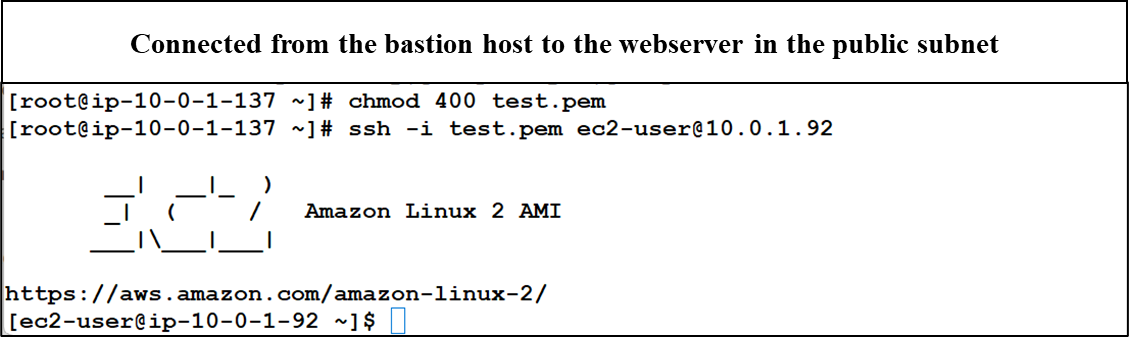
**Step 6:** connected to the appserver2 (natgateway enabled) in the private subnet, **installed nginx** and hosted the content “**welcome to the nginx server**”

**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.

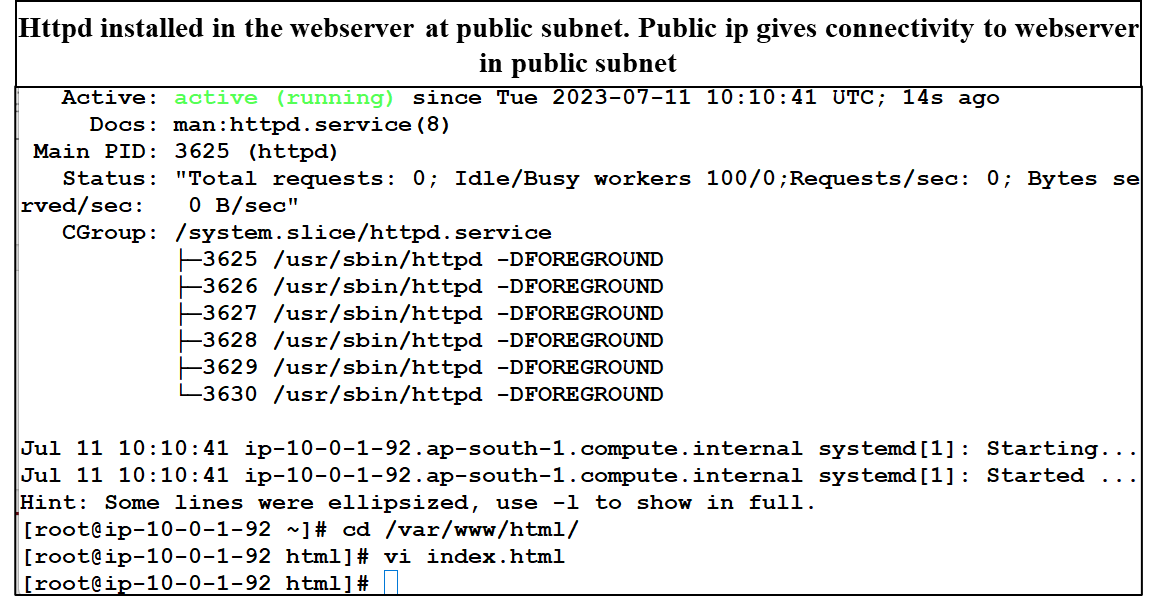
**Step3 output**

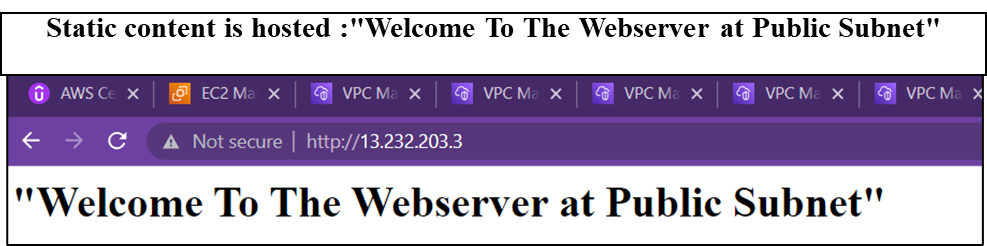
login to bastion host and connect to the webserver in the public subnet



**Step 4 output**

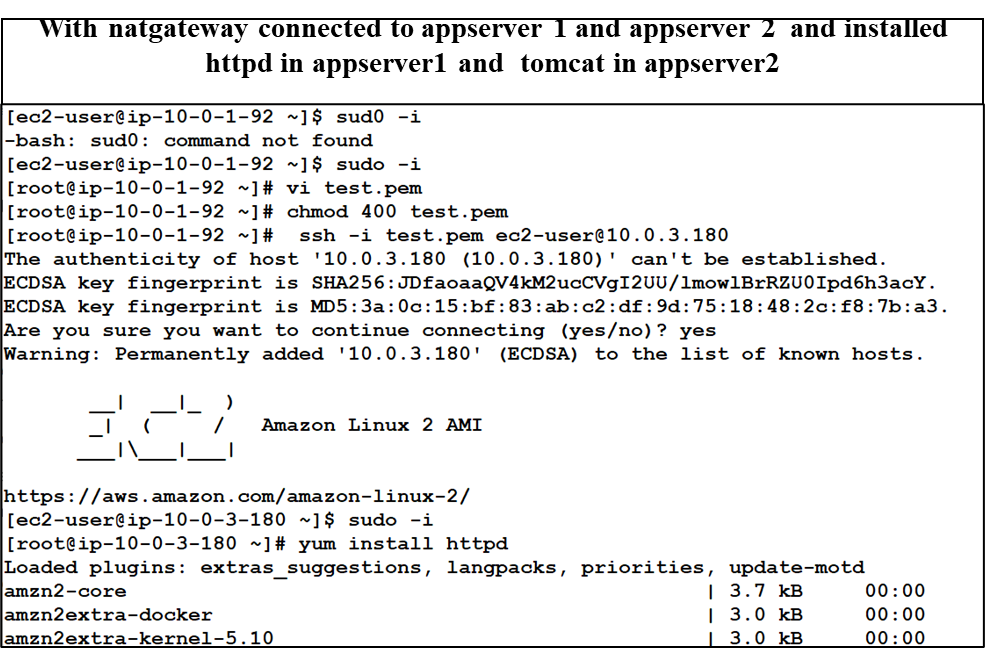
Install httpd in webserver and hosted the content “Welcome To The Webserver at Public Subnet”

****

****

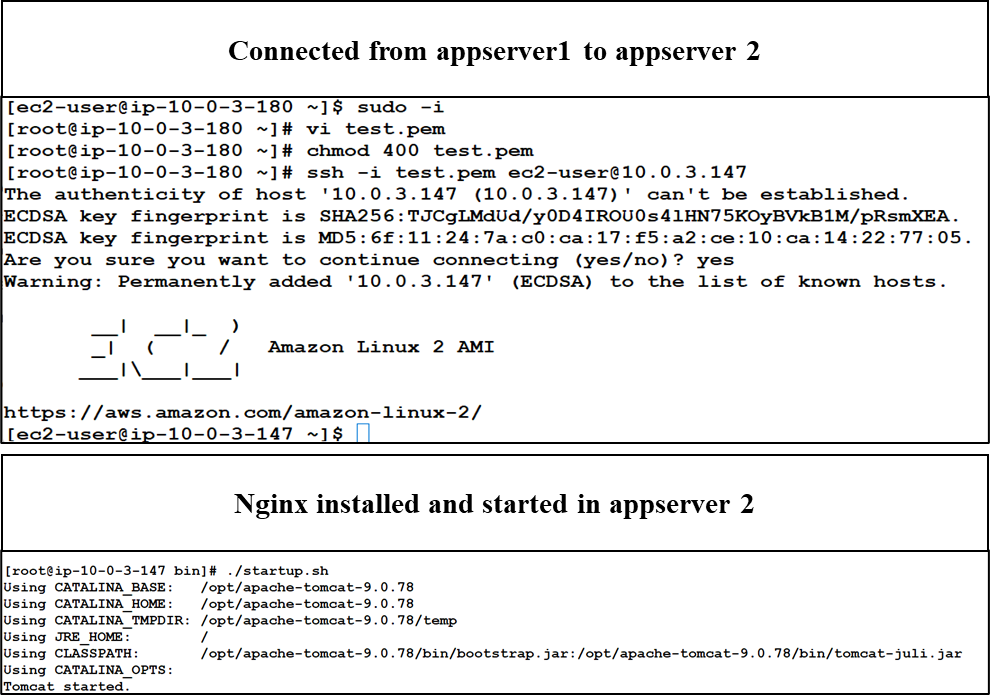
**Step 5 output :**

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

****

**Step 6 output:**

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

****

**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:**

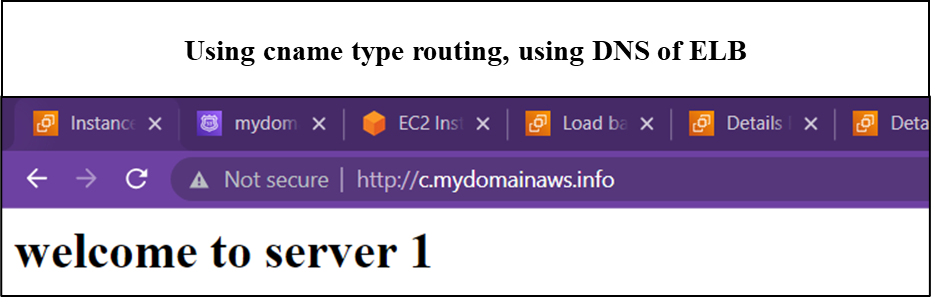
Route53 (dns) is used to create hosted zone and records with cname type routing and mapped the nameservers in the godaddy domain page. Domain name was hit in the browser to check the content of all the servers.

**ns-1731.awsdns-24.co.uk**

**ns-262.awsdns-32.com**

**ns-1519.awsdns-61.org**

**ns-1013.awsdns-62.net**



1. **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

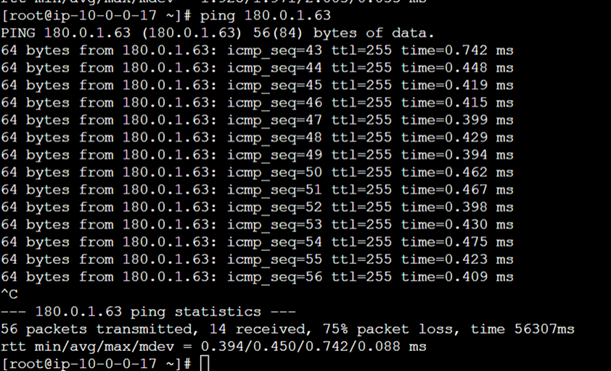
**Step3**: At VPC 2, accepted the request obtained from the VPC-1

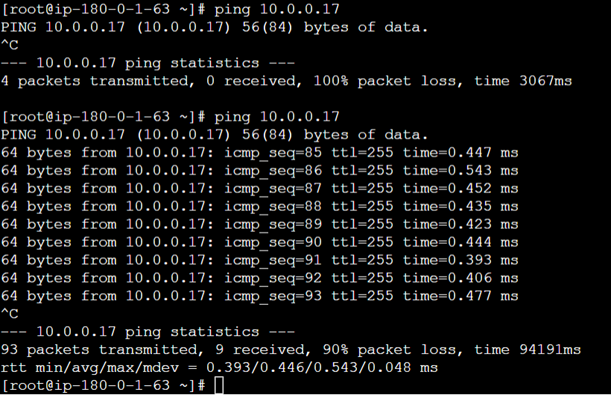
**Step 4**: At routetable-1 (vpc1), added a route for cidr:180.0.0.0/16 with peering condition and at routetable-2(vpc2), added a route for cidr:10.0.0.0/16 with peering condition.

**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.





**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

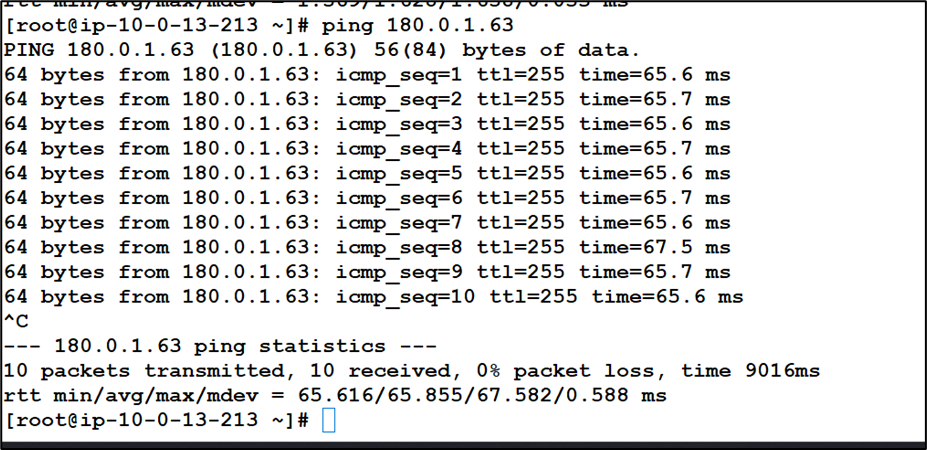
**Step 3**: At VPC 2, accepted the request obtained from the VPC-1

**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.



===============================================

**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)

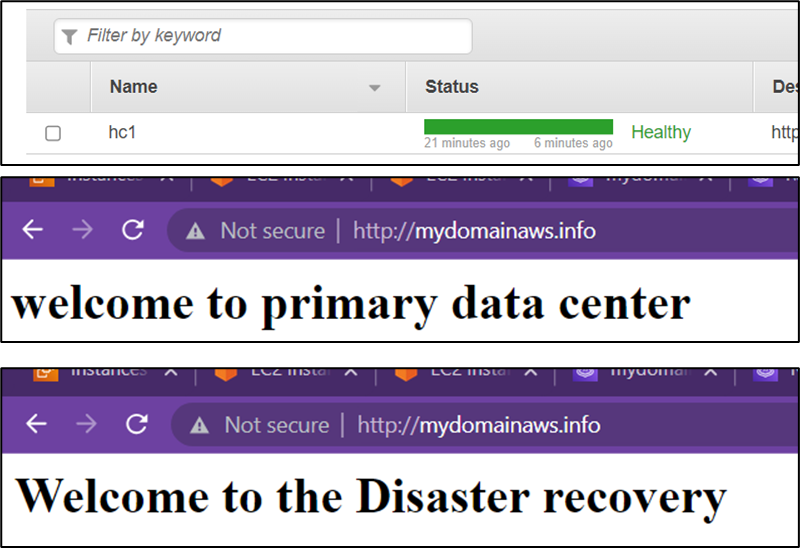
**Name servers obtained from the hosted zone and mapped with go daddy domain**

ns-830.awsdns-39.net

ns-2006.awsdns-58.co.uk

ns-114.awsdns-14.com

ns-1107.awsdns-10.org

****