# **Description**

Designing a Highly Available Network with Custom VPC

In this case, you’ll learn creation and usage of EC2, VPC, Subnet, Internet Gateways, Security Groups, Route Tables.

# **Acceptance criteria**

We are going to design the network for a highly available two tier web application. The web servers will be deployed in two public subnets across two availability zones having Internet connectivity and DB servers will be deployed in two private subnets across two availability zones. The DB servers will use network address translation (NAT) service for accessing Internet.

# **Instructions**

**Step 01 Creating a VPC**

Login to your AWS account and find VPC under Networking & Content Delivery category.

1. Click on Your VPCs in the side bar and then click on Create VPC.

## ***Points to answer before processing to next step***

***Did you notice that a VPC (default VPC) was already created?***

***Find out what other resources were automatically created for you in VPC and why?***

1. Now you need to give a name to your VPC and select a CIDR notation.

\* Name tag: MyVPC

\* IPv4 CIDR block: 10.0.0.0/16

\* IPv4 CIDR block: No IPv6 CIDR Block

\* Tenancy: default

1. Click on Yes Create.

*Select MyVPC and click on Action drop-down.*

*Ensure that Edit DNS Resolution and Edit DNS Hostnames are set to Yes.*

**Step 02 - Creating Subnets**

We would now create one public and one private subnet in both availability zones for the high availability of our resources.

Click on Subnets in the sidebar of the VPC Dashboard, click on Create Subnet

1. Name tag: MyPrivateSubnet01
2. VPC: MyVPC
3. Availability Zone: \_choose the first one you see\_
4. IPv4 CIDR block: 10.0.1.0/24

Click on Yes, Create. Your new subnet should have been created now and show up on the screen.

Repeat the same steps to create 3 more Subnets with below configuration.

1. Name tag: MyPrivateSubnet02
2. VPC: MyVPC
3. Availability Zone: \_choose the second one you see\_
4. IPv4 CIDR block: 10.0.2.0/24
5. Name tag: MyPublicSubnet01
6. VPC: MyVPC
7. Availability Zone: \_choose the first one you see\_
8. IPv4 CIDR block: 10.0.3.0/24
9. Name tag: MyPublicSubnet02
10. VPC: MyVPC
11. Availability Zone: \_choose the second one you see\_
12. IPv4 CIDR block: 10.0.4.0/24

Once all the subnets are created, select MyPublicSubnet01 and click on the Subnet Actions dropdown; go to Modify auto-assign IP settings and check Enable auto-assign public IPv4 address box.

Click on Save. Repeat the same step for MyPublicsubnet02 as well. Your four new subnets should be visible to you in subnet section similar to the below picture.

## ***Points to Answer before going to the next step***

***\_Why is the available number of IPs showing as 251, where are the rest 5 IPs used?\_***

***\_Why have we created two private and public in different subnets? Should we not create both Public subnets in one AZ and both Private in another AZ? \_***

**Step 03 - Create Internet gateway**

## ***Points to Answer before proceeding***

***As you might have noticed, there were similar steps taken in creating the Public and Private subnets, what differentiates them?***

A public subnet is the one that has a route to Internet Gateway in its routing table. So now let's create an Internet Gateway. Click on Internet Gateways in the sidebar of VPC Dashboard and then click on Create Internet Gateway.

1. Name tag: MyIGW
2. Click on Yes, Create.

You will see the state of MyIGW as detached as it is not yet attached to a VPC.

Select the MyIGW and click on Attach to VPC, select MyVPC from the drop-down in next pop up and click on Yes, Attach.

## ***Points to Answer before proceeding***

***\_Why was the default VPC not showing in the dropdown. \_***

**Step 04 - Create Route table (public) and assign to relevant Subnets**

Click on Route tables in the side bar, you should see a Route Table already created for you, assigned to MyVPC.

1. Click on Create Route Table
2. Name tag: MyPublicRoute
3. VPC: MyVPC
4. Click on Yes, Create.

A new route table would have come up now.

While the MyPublicRoute selected, click on Routes tab in the lower half of the screen. You would see that it already has an entry for local traffic. We now should add the route entry meant for Internet. Click on Edit and then on Add route. Fill in the below details in the new blank route table entry.

1. Destination: 0.0.0.0/0
2. Target: Internet Gateway (you would see the Internet gateway name in the drop down)
3. Click on Save routes.

This way we have added an entry to Internet in our public route table, now is the time to assign the route table to our public subnets.

1. Click on the Subnet Associations tab right next to the Routes tab.

## ***Points to Answer before proceeding***

***\_You would see that all four subnets that you created are associated with the main route table, why?\_***

1. Click on Edit subnet associations and select the two Public Subnets that you created.
2. Save.

**Step 05 – Security Groups**

Let us now create three different 'Security Groups' for bastion hosts, application server, database and load balancer. We would leverage them in coming labs.

In the navigation pane find and click on 'Security Groups'

1. Click on 'Create Security Group'
2. Security group name\*: My-App-SG
3. Description\*: This SG is to be used for application servers.
4. VPC: MyVPC
5. Click on Create

Create two more security groups with following configurations --

1. Security group name: My-DB-SG
2. Description: This SG is to be used for database servers.
3. VPC: MyVPC
4. Security group name: My-ALB-SG
5. Description: This SG is to be used for application load balancers.
6. VPC: MyVPC
7. Security group name: My-BastionHost-SG
8. Description: This SG is to be used for bastions hosts.
9. VPC: MyVPC

Select either of the Security Group now and click on 'Inbound Rules' tab.

Click on 'Edit Rules' and add rules for incoming traffic on the security groups like mentioned below.

1. My-BastionHost-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| RDP | TCP | 3389 | Anywhere | 0.0.0.0/0 |

1. My-App-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| HTTP | TCP | 80 | Anywhere | 0.0.0.0/0 |

| HTTPS | TCP | 443 | Anywhere | 0.0.0.0/0 |

| RDP | TCP | 3389 | Anywhere | 0.0.0.0/0 |

1. My-DB-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| MYSQL/Aurora | TCP | 3306 | Custom | \<My-App-SG> |

| RDP | TCP | 3389 | Anywhere | 0.0.0.0/0 |

1. My-ALB-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| HTTP | TCP | 80 | Anywhere | 0.0.0.0/0 |

| HTTPS | TCP | 443 | Anywhere | 0.0.0.0/0 |

## ***Points to Answer before proceeding***

***These rules are not perfect but will suffice our requirement as of now. We will be changing them in sometime. In ideal scenario these rules should look like mentioned below. Can you identify the difference?***

1. My-BastionHost-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| RDP | TCP | 3389 | Anywhere | 0.0.0.0/0 |

1. My-App-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| HTTP | TCP | 80 | Custom | \<My-ALB-SG> |

| HTTPS | TCP | 443 | Custom | \<My-ALB-SG> |

| RDP | TCP | 3389 | Custom | \<My-BastionHost-SG> |

1. My-DB-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| MYSQL/Aurora | TCP | 3306 | Custom | \<My-App-SG> |

| RDP | TCP | 3389 | Custom | \<My-BastionHost-SG> |

1. My-ALB-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| HTTP | TCP | 80 | Anywhere | 0.0.0.0/0 |

| HTTPS | TCP | 443 | Anywhere | 0.0.0.0/0 |

For now, our VPC configuration is complete. The instances launched in our public subnets should have access to Internet and the instances in our private subnet should not. We would verify the same in the next section.

**Step 06 - Creating EC2 instances**

Let us switch to EC2 Dashboard now and click on Launch Instance.

1. Amazon Machine Image: "Microsoft Windows Server 2016 Base" (free tier eligible)
2. Instance Type: t2.micro
3. Configure Instance Details: select the below mentioned points and leave everything else as default.
4. Network: MyVPC
5. Subnet: MyPublicSubnet01
6. Add Storage: Leave defaults (Your instance will come with a root volume of 30 GB as you can see in this screen. We can add additional EBS volumes if need be)
7. Add Tags

\* Key: Name

\* Value: MyAppServer

1. Configure Security Group: Select existing -> My-App-SG
2. Click on Review and Launch.

On the next page ensure that your AMI is free tier eligible and Instance Type is showing as t2.micro.

1. Click on Launch.
2. On the next window, create a new Key Pair, Key pair name: mykey and then Download Key pair.
3. Finally click on Launch Instance

We have just created an EC2 instance in our public subnet, now we would create an EC2 instance in private subnet following similar steps.

Go back to EC2 Dashboard now and click on Launch Instance.

1. Amazon Machine Image: "Microsoft Windows Server 2016 Base" (free tier eligible)
2. Instance Type: t2.micro
3. Configure Instance Details: select the below mentioned points and leave everything else as default.

\* Network: MyVPC

\* Subnet: MyPrivateSubnet01

1. Add Storage: Leave defaults (Your instance will come with a root volume of 30 GB as you can see in this screen. We can add additional EBS volumes if need be)
2. Add Tags

\* Key: Name

\* Value: MyDBServer

1. Configure Security Group: Select existing -> My-DB-SG
2. Click on Review and Launch.

On the next page check that your AMI is free tier eligible and Instance Type is showing as t2.micro.

1. Click on Launch.
2. On the next window, select to use existing Key Pair 'mykey'.
3. Click on Launch Instance

Go back to your EC2 instance page. You should see your two instances.

## **Points to Answer before proceeding**

**\_Did you notice that your MyAppServer has got a public IP and public DNS while MyDBServer has not, why? \_**

**\_Why are both are running in the same AZ? \_**

**Step 07 - Verifying the connectivity**

We now have created two EC2 instances one in each public and private subnet. We would now verify whether our network configuration is working as desired.

Let us RDP to the MyAppServer.

1. Select the MyAppServer in the dashboard and click on 'Connect'
2. You now have to mention the path of your 'key pair' and decrypt the windows password.
3. Get the login credentials and login to the instance.
4. Once you are logged into your MyAppServer EC2 instance, you may verify if the machine can reach Internet.

In the above step you connected to your EC2 instance in public subnet and verified that it had Internet connectivity. Our next step would be login to the EC2 instance launched in the private subnet and verify that it should not have Internet connectivity.

We will use our public EC2 instance as a bastion host (jump box) to login to the instance in the private subnet. In ideal scenario we should have created another EC2 instance as bastion host, we are using the 'MyAppServer' as jump server just to minimize cost and to be in free tier limits as long as possible in the lab exercise.

1. Retrieve the password for the private instance using the AWS management console and initiate an RDP session from within the public instance.

Let us quickly try to check to see if this EC2 instance can reach Internet.

The easiest way is by pinging google.com

In all possibilities, the connection should not work. This proves that your MyDBServer in MyPrivatesubnet01 does not have direct access to Internet. Keep the session opened. In the next steps, we would enable NATing service for private subnets to give one-way Internet access.

1. Go back to your VPC dashboard.
2. Click on NAT Gateways in the sidebar of VPC Dashboard and then click on Create NAT Gateway.
3. Select the following configurations on the next page.
4. Subnet: MyPublicSubnet01 (select from the dropdown)
5. Elastic IP Allocation ID: Create New EIP
6. Click on Create a NAT Gateway.

## ***Points to Answer before proceeding***

***\_Why have you created this NAT Gateway in a public subnet? \_***

***\_Why did you select MyPublicSubnet01 and not MyPublicSubnet02? \_***

Now as your NAT Gateway has been created, we will add this in the private route table.

1. Go to Route Tables, create a new route table "MyPrivateRoute" and assign it to the two Private Subnets (Follow the similar process as you did in activity 3.

As expected, you would see that it has an entry for local traffic. We now should add the route entry meant for Internet.

1. Click on Edit and then on Add Another Route. Fill in the below details in the new blank route table entry.
2. Destination: 0.0.0.0/0
3. Target: NAT Gateway (Select the one you created)
4. Click on save.

So, you have now created a NAT Gateway which is a managed service by Amazon and assigned it to the route table which is assigned to your private subnets. This way the EC2 instances created in private subnets would get the outbound access to Internet for downloading updates/patches etc.

Let us verify the same by going back and accessing Internet from the instance in private subnet. It should work now if you have followed the steps carefully.

If the objectives have successfully completed, then take the below snapshots -

1 - Of your RDP session showing the IP address of the private instance and working internet.

2 - The snapshot of the NAT Gateway assignment to the private route table.

You can upload these in git, attach them while submitting this case study completion.

**Step 08 - Clean up**

Let's clean up. Follow the order or you will get dependency errors.

1. Terminate both the instances.
2. Delete NAT Gateway (deleting NAT Gateway might take a minute or two, keep refreshing the page till the time you see it is deleted)
3. Release the Elastic IP that was created for your NAT Gateway (it will not be released until the NAT gateway is deleted).

***The NAT Gateway is a chargeable resource and typically it is ~USD 0.05/INR 5 to 6 per hour, ensure you delete it within an hour of creation. You would need to pay this amount once AWS sends you the bill at the end of month. Elastic IPs are chargeable resources if they are lying unused, there will be no fee as long as you delete it post NAT Gateway deletion.***

Leave the other resources. They are free and we will leverage them in subsequent labs.