Please use your free tier AWS account to complete this lab

Designing a Custom VPC with High Availability

VPC Lab – Part 01 of 02

In this lab we are going to design the **network** for a highly available web application. The web servers will be deployed across two availability zones having internet connectivity and app/DB servers will be deployed in the private subnets, the app/DB servers will use network address translation (NAT) service for accessing internet.

Ensure that you are in Mumbai region.

Activity 01 – Creating a VPC

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

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

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!

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

Name tag: MyVPC

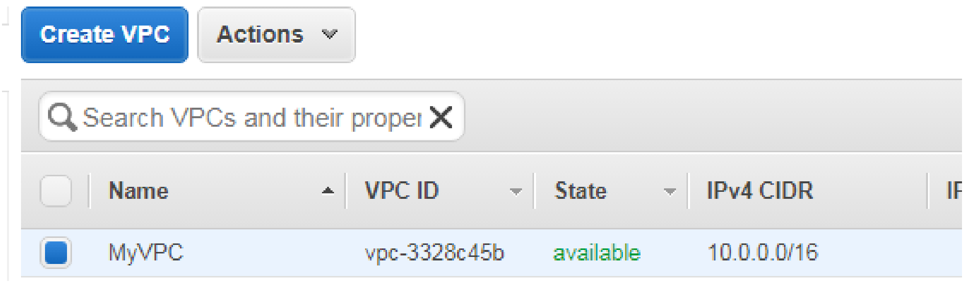
IPv4 CIDR block: 10.0.0.0/16

IPv4 CIDR block: No IPv6 CIDR Block

Tenancy: default

Click on Yes Create

You should now see your VPC created similar to below picture.



Name	VPC ID	State	IPv4 CIDR
MyVPC	vpc-3328c45b	available	10.0.0.0/16

Select MyVPC and click on Action dropdown. Ensure that Edit DNS Resolution and Edit DNS Hostnames are set to Yes.

Activity 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

Name tag: MyPrivateSubnet01

VPC: MyVPC

Availability Zone: ap-south-1a

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.

Name tag: MyPrivateSubnet02

VPC: MyVPC

Availability Zone: ap-south-1b

IPv4 CIDR block: 10.0.2.0/24

Name tag: MyPublicSubnet01

VPC: MyVPC

Availability Zone: ap-south-1a

IPv4 CIDR block: 10.0.3.0/24

Name tag: MyPublicSubnet02

VPC: MyVPC

Availability Zone: ap-south-1b

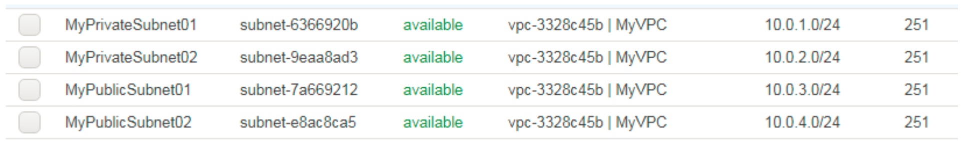
IPv4 CIDR block: 10.0.4.0/24

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.

You four new subnets should be visible to you in subnet section similar to the below picture.



☐	MyPrivateSubnet01	subnet-6366920b	available	vpc-3328c45b | MyVPC	10.0.1.0/24	251
☐	MyPrivateSubnet02	subnet-9eaa8ad3	available	vpc-3328c45b | MyVPC	10.0.2.0/24	251
☐	MyPublicSubnet01	subnet-7a669212	available	vpc-3328c45b | MyVPC	10.0.3.0/24	251
☐	MyPublicSubnet02	subnet-e8ac8ca5	available	vpc-3328c45b | MyVPC	10.0.4.0/24	251

#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?

Activity 03 - Create Internet gateway

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 it's 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.

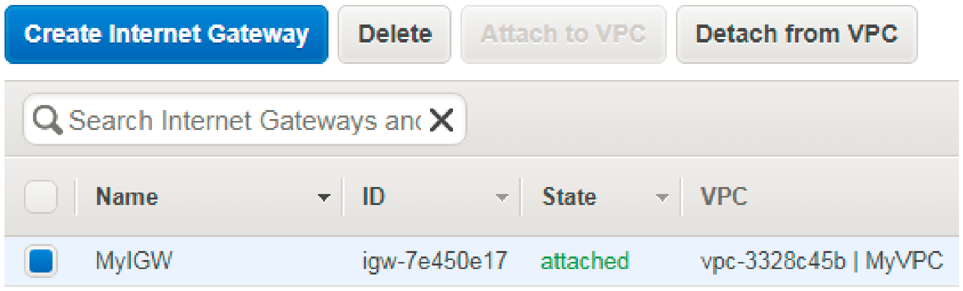
Mention Name tag: MyIGW and click on Yes, Create.

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

Select the MyIGW and click on Attache to VPC, select MyVPC from the dropdown in next pop up and click on Yes, Attach.

#Why was the default VPC not showing in the dropdown.

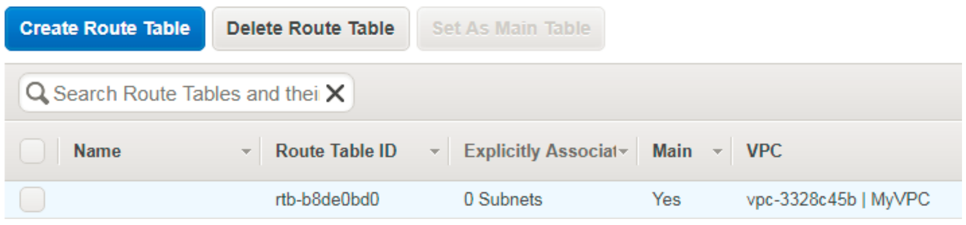
Your screen should now show like the below picture.



☐	Name	ID	State	VPC
☑	MyIGW	igw-7e450e17	attached	vpc-3328c45b | MyVPC

Activity 03 - 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 like the below picture.



Name	Route Table ID	Explicitly Associat	Main	VPC
	rtb-b8de0bd0	0 Subnets	Yes	vpc-3328c45b | MyVPC

Click on Create Route Table, Name tag: MyPublicRoute, VPC: MyVPC and click on Yes, Create. A new route table would have come up now.

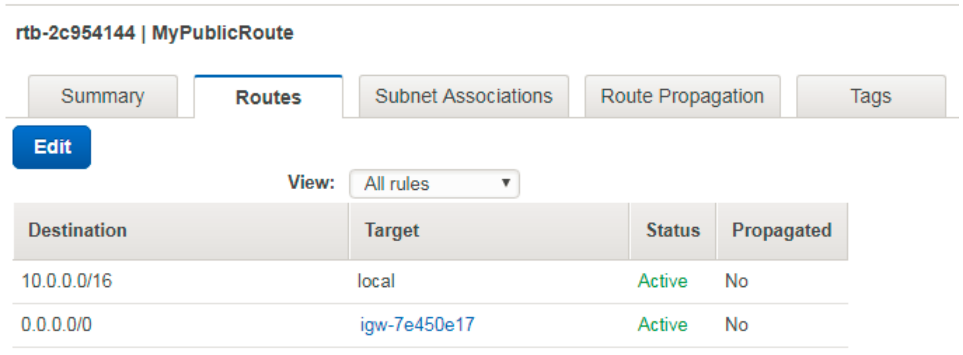
While the MyPublicRoute selected, click on Routes tab in the lower half of the screen. You would see that it already has an entry for local traffic. We now should add the route entry meant for internet.

Click on Edit and then on Add Another Route. Fill in the below details in the new blank route table entry.

Destination: 0.0.0.0/0

Target: IGW (you would see the internet gateway name in the drop down when you click in this blank field)

Click on save, your configuration should look like below picture.



View: All rules

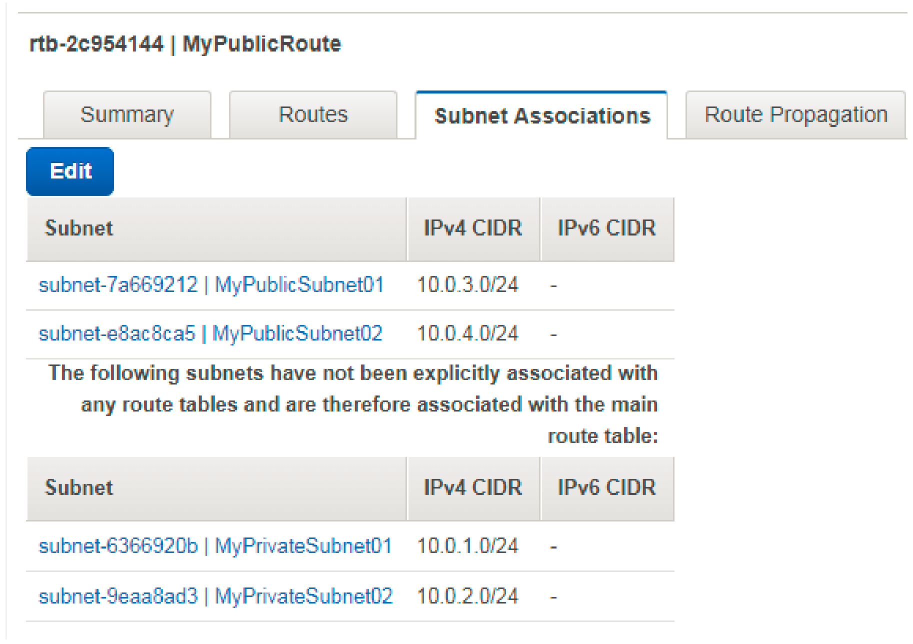
Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-7e450e17	Active	No

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.

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

#You would see that all four subnets that you created are associated with the main route table, why?

Click on Edit and select the two Public Subnets that you created. The subnet association tab should now look like below picture.



rtb-2c954144 | MyPublicRoute

Subnet	IPv4 CIDR	IPv6 CIDR
subnet-7a669212 | MyPublicSubnet01	10.0.3.0/24	-
subnet-e8ac8ca5 | MyPublicSubnet02	10.0.4.0/24	-

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Subnet	IPv4 CIDR	IPv6 CIDR
subnet-6366920b | MyPrivateSubnet01	10.0.1.0/24	-
subnet-9eaa8ad3 | MyPrivateSubnet02	10.0.2.0/24	-

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 n the next section.

VPC Lab – Part 02 of 02

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

Select the “Microsoft Windows Server 2016 Base”

Select the t2.micro instance type and click on Next.

On the Configure Instance Details page select the below mentioned points and leave everything else as default.

Network: MyVPC

Subnet: MyPrivateSubnet01

Click on Next: Add Storage

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, as of now click on Next: Add Tags

Create a Tag with Key: Name and value: MyPrivateInstance01

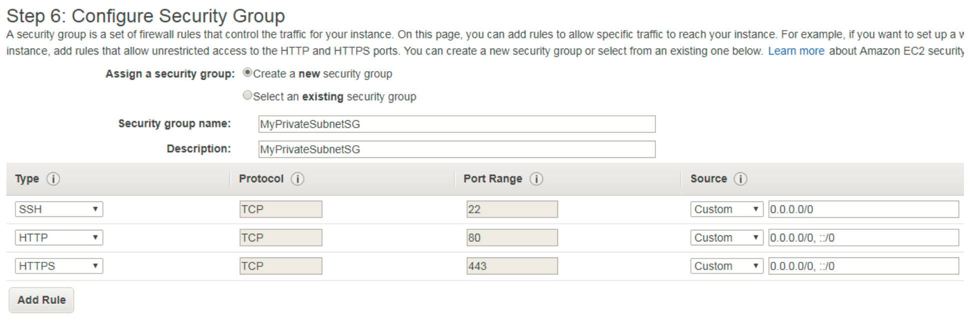
Click on Next: Configure Security Group

Click on the Create a new security group and give MyPrivateSubnetSG in both the fields of Security Group name as well as Description.

#do you see the auto populated entry for RDP, why?

Now click on Add Rule and add an entry of HTTP and HTTPS. Leave every other field as it is.

Your screen should show like the below picture.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a v
instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: MyPrivateSubnetSG

Description: MyPrivateSubnetSG

Type	Protocol	Port Range	Source	
SSH	TCP	22	Custom	0.0.0.0/0
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0

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.

Click on Launch.

On the next window, create a new Key Pair, Key pair name: mykey and then Download Key pair.

Finally click on Launch Instance

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

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

Select the “Microsoft Windows Server 2016 Base”

Select the t2.micro instance type and click on Next.

On the Configure Instance Details page select the below mentioned points and leave everything else as default.

Network: MyVPC

Subnet: MyPublicSubnet01

Click on Next: Add Storage

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, as of now click on Next: Add Tags

Create a Tag with Key: Name and value: MyPublicInstance01

Click on Next: Configure Security Group

Click on the Create a new security group and give MyPublicSubnetSG in both the fields of Security Group name as well as Description.

Now click on Add Rule and add an entry of HTTP and HTTPS. Leave every other field as it is.

Click on Review and Launch.

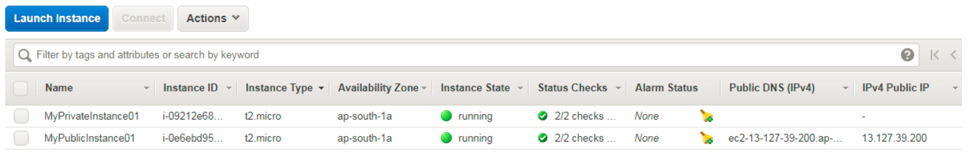
On the next page ensure that your AMI is free tier eligible and Instance Type is showing as t2.micro.

Click on Launch.

On the next window, choose an existing Key Pair, Key pair name: mykey and then proceed.

Click on Launch Instance

Go back to your EC2 instance page. The page should look like the below picture.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
MyPrivateInstance01	i-09212e68...	t2.micro	ap-south-1a	● running	✓ 2/2 checks ...	None		-
MyPublicInstance01	i-0e6ebd95...	t2.micro	ap-south-1a	● running	✓ 2/2 checks ...	None	ec2-13-127-39-200.ap-...	13.127.39.200

#Did you notice that your MyPublicInstance01 has got a public IP and public DNS while MyPrivateInstance01 has not, why?

#Why are both are running in the same AZ?

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 public instance

Once you are logged into your MyPublicInstance01 EC2, 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 jump box to login to the instance in the private subnet.

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

Let us quickly try to check over the browser if we this EC2 instance can reach internet.

In all possibilities, the connection should not work. This proves that your MyPrivateInstance01 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.

Go back to your VPC dashboard.

Click on NAT Gateways in the sidebar of VPC Dashboard and then click on Create NAT Gateway.

Select the following configurations on the next page.

Subnet*: MyPublicSubnet01 (select from the dropdown)

Elastic IP Allocation ID*: Create New EIP

Now click on Create a NAT Gateway.

#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.

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.

Click on Edit and then on Add Another Route. Fill in the below details in the new blank route table entry.

Destination: 0.0.0.0/0

Target: NAT Gateway (you would see the internet gateway as well as the NAT Gateway name in the drop down when you click in this blank field)

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 through browser.

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 lab completion on TopGear.

Let’s clean up.

Terminate both the instances.

Go back to your VPC Dashboard and NAT Gateways

Select your NAT Gateway --> Action --> Delete NAT Gateway

Deleting NAT Gateway might take a minute or two, keep refreshing the page till the time you see it is deleted.

Click on the Elastic IPs from the sidebar and release the Elastic IP that was created for your NAT Gateway.

Select the Elastic IP --> Actions --> Release Addresses --> Release

You should find the one that is not assigned to any resources currently, in most probable cases you would only see one here considering you are doing this Lab.

Now select your MyVPC and action - delete.

Lab Complete.