# Designing and Building a Custom VPC from Scratch

## Introduction

This hands-on lab provides you with some experience building and connecting the following services inside AWS: VPC, subnets, internet gateway, NAT gateways, Bastion host, route tables, security groups, and network access control lists (NACLs). These services are the foundation of networking architecture inside of AWS and cover concepts such as infrastructure, design, routing, and security.

## Solution

Log in to the live AWS environment using the credentials provided. Make sure you're in the N. Virginia (us-east-1) region throughout the lab.

### Create VPC and Subnet Architecture

#### Create a VPC

1. Navigate to **VPC** > **Your VPCs**.
2. Click **Create VPC**, and set the following values:
   * Name tag: **labVPC**
   * IPv4 CIDR block: **10.0.0.0/16**
   * IPv6 CIDR block: **Amazon provided IPv6 CIDR block**
3. Leave the Tenancy field as its default value.
4. Click **Create**.

#### Create Subnets

##### publicA Subnet

1. Click **Subnets** in the left-hand menu.
2. Click **Create subnet**.
   * Name tag: **publicA**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1a**
   * IPv4 CIDR block: **10.0.0.0/24**
3. Click **Create**, and close out of the success message.

##### publicB Subnet

1. Click **Create subnet**.
   * Name tag: **publicB**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1b**
   * IPv4 CIDR block: **10.0.1.0/24**
2. Click **Create**, and close out of the success message.

##### publicC Subnet

1. Click **Create subnet**.
   * Name tag: **publicC**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1c**
   * IPv4 CIDR block: **10.0.2.0/24**
2. Click **Create**, and close out of the success message.

##### privateA Subnet

1. Click **Create subnet**.
   * Name tag: **privateA**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1a**
   * IPv4 CIDR block: **10.0.4.0/24**
2. Click **Create**, and close out of the success message.

##### privateB Subnet

1. Click **Create subnet**.
   * Name tag: **privateB**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1b**
   * IPv4 CIDR block: **10.0.5.0/24**
2. Click **Create**, and close out of the success message.

##### privateC Subnet

1. Click **Create subnet**.
   * Name tag: **privateC**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1c**
   * IPv4 CIDR block: **10.0.6.0/24**
2. Click **Create**, and close out of the success message.

##### dbA Subnet

1. Click **Create subnet**.
   * Name tag: **dbA**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1a**
   * IPv4 CIDR block: **10.0.8.0/24**
2. Click **Create**, and close out of the success message.

##### dbB Subnet

1. Click **Create subnet**.
   * Name tag: **dbB**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1b**
   * IPv4 CIDR block: **10.0.9.0/24**
2. Click **Create**, and close out of the success message.

##### dbC Subnet

1. Click **Create subnet**.
   * Name tag: **dbC**
   * VPC: **labVPC**
   * Availability Zone: **us-east-1c**
   * IPv4 CIDR block: **10.0.10.0/24**
2. Click **Create**, and close out of the success message.

### Create Internet Gateway, Public Routing, and Bastion Host

1. Select **publicA**, and click **Actions** > **Modify auto-assign IP settings**.
2. Check the box to **Enable auto-assign public IPv4 address**.
3. Click **Save**, and then un-select **publicA**.
4. Select **publicB**, and click **Actions** > **Modify auto-assign IP settings**.
5. Check the box to **Enable auto-assign public IPv4 address**.
6. Click **Save**, and then un-select **publicB**.
7. Select **publicC**, and click **Actions** > **Modify auto-assign IP settings**.
8. Check the box to **Enable auto-assign public IPv4 address**.
9. Click **Save**.

#### Configure Internet Gateway

1. Click **Internet Gateways** in the left-hand menu.
2. Click **Create internet gateway**.
3. Set the name tag as "labVPCIGW", and click **Create**.
4. Select the newly created IGW, and click **Actions** > **Attach to VPC**.
5. Select **labVPC**, and click **Attach**.

#### Configure Routing

1. Click **Route Tables** in the left-hand menu.
2. Click **Create route table**, and set the following values:
   * Name tag: **publicRT**
   * VPC: **labVPC**
3. Click **Create**.

#### Add Default Public Route

1. Select **publicRT**, and click the **Routes** tab.
2. Click **Edit routes**, **Add route**, and set the following values:
   * Destination: **0.0.0.0/0**
   * Target: **Internet Gateway**, and select **labVPCIGW**
3. Click **Add route** again, set the following values:
   * Destination: **::/0**
   * Target: **Internet Gateway**, and select **labVPCIGW**
4. Click **Save routes**.
5. Click **Close**.

#### Associate with Subnets

1. Select **publicRT**, and click the **Subnet Associations** tab.
2. Click **Edit subnet associations**.
3. Select **publicA**, **publicB**, and **publicC**.
4. Click **Save**.

#### Create a Bastion Host

Since these subnets are public, then, in theory, anything we deploy into them should be publicly accessible. Now, we'll create a bastion host, which is a way you can access a secure VPC from outside — meaning, we can connect to the bastion host via SSH and use it to connect into the VPC.

1. Navigate to **EC2** > **Instances**.
2. Click **Launch Instance**.
3. On the AMI page, select the Amazon Linux 2 AMI with 64-bit (x86) architecture.
4. Choose the **t3.micro** instance type, and click **Next: Configure Instance Details**.
5. On the Configure Instance Details page, set the following values:
   * Network: **labVPC**
   * Subnet: **publicB**
   * Auto-assign Public IP: **Use subnet setting (Enable)**
6. Click **Next: Add Storage**, and then click **Next: Add Tags**.
7. On the Add Tags page, add the following tag:
   * Key: **Name**
   * Value: **BastionHost**
8. Click **Next: Configure Security Group**.
9. Select **Create a new security group**, and set the following values:
   * Security group name: **bastionSG**
   * Description: **bastionSG**
10. Click **Review and Launch**, and then **Launch**.
11. In the key pair dialog, select **Create a new key pair**.
12. Give it a Key pair name of "vpclab".
13. Click **Download Key Pair**, and then **Launch Instances**.
14. Click **View Instances**, and give it a few minutes to enter the running state.

#### Verify Bastion Host Is Working

1. When the bastion host has 2/2 status checks, select the instance, click **Connect**, and copy the ssh connection command.
2. Open a terminal window.

**Note:** Windows users can connect to the instance using [this as a guide](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html).

1. Change to your downloads folder, where the key pair file is saved:

cd Downloads

1. Change permissions on the key pair file:

chmod 400 vpclab.pem

1. Run the ssh connection command you copied from the AWS console to connect to your bastion host.
2. Enter yes at the prompt.

### Configure Private Internet Connectivity Using NAT Gateway

#### Create the NAT Gateways

1. In the AWS console, navigate to **VPC** > **NAT Gateways**.
2. Click **Create NAT Gateway**.
3. Set the subnet to **publicA**.
4. Click **Allocate Elastic IP address** and then **Create a NAT Gateway**.
5. Click **Close**.
6. Click **Create NAT Gateway**.
7. Set the subnet to **publicB**.
8. Click **Allocate Elastic IP address** and then **Create a NAT Gateway**.
9. Click **Close**.
10. Click **Create NAT Gateway**.
11. Set the subnet to **publicC**.
12. Click **Allocate Elastic IP address** and then **Create a NAT Gateway**.
13. Click **Close**.
14. Select each NAT gateway, and copy each one's NAT Gateway ID as well as the public subnet it's in (this information can be found in the Details tab). Paste these values into a text file, as we will need them later.

#### Create Three Private Route Tables

1. Click **Route Tables**.
2. Click **Create route table**, and set the following values:
   * Name tag: **privateA-RT**
   * VPC: **labVPC**
3. Click **Create** and then **Close**.
4. Click **Create route table**, and set the following values:
   * Name tag: **privateB-RT**
   * VPC: **labVPC**
5. Click **Create** and then **Close**.
6. Click **Create route table**, and set the following values:
   * Name tag: **privateC-RT**
   * VPC: **labVPC**
7. Click **Create** and then **Close**.

#### Route Table Associations

##### privateA-RT

1. With **privateA-RT** selected, click the **Subnet Associations** tab.
2. Click **Edit subnet associations**.
3. Select **dbA** and **privateA**
4. Click **Save**.
5. On the same route table, click the **Routes** tab.
6. Click **Edit routes**, **Add route**, and set the following values:
   * Destination: **0.0.0.0/0**
   * Target: **NAT Gateway**, and paste the NAT Gateway ID for the one in publicA in the list you made earlier
7. Click **Save routes** and **Close**.

##### privateB-RT

1. Select **privateB-RT**, and click the **Subnet Associations** tab.
2. Click **Edit subnet associations**.
3. Select **dbB** and **privateB**
4. Click **Save**.
5. On the same route table, click the **Routes** tab.
6. Click **Edit routes**, **Add route**, and set the following values:
   * Destination: **0.0.0.0/0**
   * Target: **NAT Gateway**, and paste the NAT Gateway ID for the one in publicB in the list you made earlier
7. Click **Save routes** and **Close**.

##### privateC-RT

1. Select **privateC-RT**, and click the **Subnet Associations** tab.
2. Click **Edit subnet associations**.
3. Select **dbC** and **privateC**
4. Click **Save**.
5. On the same route table, click the **Routes** tab.
6. Click **Edit routes**, **Add route**, and set the following values:
   * Destination: **0.0.0.0/0**
   * Target: **NAT Gateway**, and paste the NAT Gateway ID for the one in publicC in the list you made earlier
7. Click **Save routes** and **Close**.

### Configure and Test VPC Security

1. Navigate to EC2.
2. Click **Launch instance**.
3. On the AMI page, select the Amazon Linux 2 AMI with 64-bit (x86) architecture.
4. Choose the **t3.micro** instance type, and click **Next: Configure Instance Details**.
5. On the Configure Instance Details page, set the following values:
   * Network: **labVPC**
   * Subnet: **privateA**
   * Auto-assign Public IP: **Use subnet setting (Disable)**
6. Click **Next: Add Storage**, and then click **Next: Add Tags**.
7. On the Add Tags page, add the following tag:
   * Key: **Name**
   * Value: **appserver**
8. Click **Next: Configure Security Group**.
9. Select **Create a new security group**, and set the following values:
   * Security group name: **appSG**
   * Description: **appSG**
10. Change the rule Source to **bastionSG**.
11. Click **Review and Launch**, and then **Launch**.
12. In the key pair dialog, select **Choose an existing key pair**.
13. Choose the **vpclab** key pair.
14. Click **Launch Instances**.
15. Click **View Instances**, and give it a few minutes to enter the running state.

#### Use SSH Key Forwarding

Now, we're going to use a special feature of SSH that allows us to forward keys. We're going to connect to the bastion host using SSH and use this forwarding feature to allow us to SSH from the bastion host to the app server without having to have the SSH key also on the bastion host. This is more efficient, secure, and saves us a step.

**Note:** Windows users, when using [SSH Key Forwarding](https://aws.amazon.com/blogs/security/securely-connect-to-linux-instances-running-in-a-private-amazon-vpc/), you will need specific configuration (includes PuTTY).

1. In the terminal session, exit out of the current SSH session:

exit

1. Change to your downloads folder:

cd Downloads

1. We need to add the key to the SSH agent, enabling the key to be stored in memory and be used as part of the pass-through architecture so we can hop via the bastion host into the appserver instance. To do so, run the following:

ssh-add -K vpclab.pem

1. In the AWS console, right-click the BastionHost instance, and click **Connect**.
2. Copy the ec2-user@IP\_ADDRESS portion of the connection command.
3. In the terminal session, run the following (replacing <ec2-user@IP\_ADDRESS> with what you just copied):

ssh -A <ec2-user@BASTIONHOST\_IP\_ADDRESS>

The -A tells SSH to use forwarding.

1. Confirm you have public internet connection:

ping 1.1.1.1

We should see we're connected. Hit **Ctrl**+**C** to stop the ping.

1. In the AWS console, right-click the appserver instance, and click **Connect**.
2. Copy the ec2-user@IP\_ADDRESS portion of the connection command.
3. In the terminal session, run the following (replacing <ec2-user@IP\_ADDRESS> with what you just copied):

ssh <ec2-user@IP\_ADDRESS>

1. Enter yes at the prompt.
2. Confirm you have public internet connection:

ping 1.1.1.1

We should see we're connected, which means our NAT gateway in publicA is working. Hit **Ctrl**+**C** to stop the ping.

1. Enter the following twice to exit out of both the BastionHost and app server:

exit

#### Modify NACL

1. In the AWS console, navigate to **VPC** > **Network ACLs**.
2. With the default NACL selected, click the **Inbound Rules** tab.
3. Click **Edit inbound rules**.
4. Click **Add Rule**, and set the following values:
   * Rule #: **50**
   * Type: **ALL Traffic**
   * Protocol: **All**
   * Port Range: **All**
   * Source: Your IP address (which you can get by googling "what is my IP" in a new browser tab), and append **/32** at the end
   * Allow / Deny: **DENY**
5. Click **Save**.
6. In the terminal session, try to log in to the bastion host:

ssh -A <ec2-user@BASTIONHOST\_IP\_ADDRESS>

You won't be able to since your IP address is matched against the explicit DENY rule. Exit out of the command by hitting **Ctrl**+**C**.

1. In the AWS console, select **Edit Inbound Rules**.
2. Here, find rule #50, and using the **Delete** button located to the right of the rule's row, remove the rule.
3. Select **Save**.
4. In the terminal, try connecting to the bastion host again with:

ssh -A <ec2-user@BASTIONHOST\_IP\_ADDRESS>

This time, we connect, as the deny rule has been removed.

## Conclusion

Congratulations on completing this hands-on lab!