Linux Academy

Hands-on Lab

Building a More Secure Application with a Bastion Host and NAT Gateway

Contents

Creating a VPC 1

Web Servers and Bastion Host 2

- Create Private Instances 2
- Create Bastion Host 2
- Access the Instances 3

NAT Gateway 3

Elastic Load Balancer 4

Lab Connection Information

- Labs may take up to five minutes to build
- Access to an AWS Console is provided on the Hands-on Lab page, along with your login credentials
- Ensure you are using the N. Virginia region
- Labs will automatically end once the alloted amount of time finishes

Related Courses

AWS Certified Solutions Architect - Associate

Related Videos

Serving Traffic to and from Private Servers

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In this lab, we create secure architecture from the ground up, starting with creating a VPC, four subnets – two private and two public – across two Availability Zones, two private EC2 instances to work as web servers, a Bastion host to access these servers, and an Elastic Load Balancer to distribute traffic to our web instances.

Creating a VPC

Log in to the AWS Console using the credentials provided on the Hands-on Lab page.

Navigate to the **VPC Dashboard** and press **Your VPCs**, **Create VPC**. We gave the VPC a **Name tag** of *Security-VPC*, and set the **IPv4 CIDR block** to *10.0.0.0/16*. Leave the rest as default and press **Yes, Create**.

Click **Internet Gateways** in the left menu. Press **Create Internet Gateway** and give it a **Name tag** of *Security-IGW*. **Yes, Create**. Select the gateway and press **Attach to VPC**. Select the *Security-VPC* and press **Yes, Attach**.

Navigate to **Subnets** on the left menu. We want to create four subnets. Press **Create Subnet**. We want to give our first subnet a **Name tag** of *Public1* and associate it with our *Security-VPC*. Set the **Availability Zone** to *us-east-1a* and the **IPv4 CIDR block** to *10.0.1.0/24*. Select **Yes, Create**.

Create another subnet with a **Name tag** of *Public2* in our *Security-VPC*. Set the **Availability Zone** to *us-east-1b* and the **IPv4 CIDR block** to *10.0.2.0/24*. Select **Yes, Create**.

Create a third subnet called *Private3* in our *Security-VPC*. Set the **Availability Zone** to *us-east-1a* with the **IPv4 CIDR block** of *10.0.3.0/24*. Select **Yes, Create**.

Finally, create a fourth subnet called *Private4* in our *Security-VPC*. Set the **Availability Zone** to *us-east-1b* with the **IPv4 CIDR block** of *10.0.4.0/24*. Select **Yes, Create**.

Go to **Network ACLs** in the left menu. Select our VPC from the list and confirm that both Inbound and Outbound Rules are set to allow all.

We now want to add route tables. Click **Route Tables** in the left menu, then **Create Route Table**. Give the first route table a **Name tag** of *Security-PublicRT* and ensure is it associated with the *Security-VPC*. Press **Yes, Create**.

With the route table selected, click the **Routes** tab, and **Edit** the existing routes. **Add another route** to the **Destination** of *0.0.0.0/0* with a **Target** of our *Security-IGW*. **Save**.

Now, click the **Subnet Associations** tab and associate both *Public* subnets. **Save**.

Press **Create Route Table** to create our second table with a **Name tag** of *Security-PrivateRT* and associated with the *Security-VPC*. **Yes, Create**.

Since we only need internal routing for the route table, we do not need to add anything to the **Routes** tab.

Instead, select **Subnet Associations** and add both *Private* subnets. **Save**.

Web Servers and Bastion Host

Create Private Instances

Navigate to the **EC2 Dashboard** and press **Launch Instance**. **Select** the *Amazon Linux AMI* and leave the instance type as *t2.micro*. Press **Next: Configure Instance Details**.

Set the **Network** to the *Security-VPC* and select the *Private3* subnet. Since this is a private instance, we do not need to assign a public IP address, nor do we need to make any additional changes. **Next: Add Storage**.

Proceed through the storage and tags page, making no changes, until reaching the Security Group page. Press **Add Rule** to add an *HTTP* rule with a **Source** of *0.0.0.0/0*. Set the **Security Group name** to *Private-EC2*. **Review and Launch**.

Review the instance's setting, then press **Launch**. When prompted, select *Create a new key pair* and set the **Key pair name** to *security*. **Download Key Pair**. **Launch Instances**.

Repeat this process to create a second private instance, placing the instance in the *Private4* subnet and using the security group and key pair we made in the previous steps.

Create Bastion Host

Since neither of these instances are accessible from the internet, we need to set up a Bastion from which we can connect.

Press **Launch Instance** and **Select** the *Amazon Linux AMI*, leaving the instance type as *t2.imcro*. **Next: Configure Instance Details**.

Set the **VPC** to the *Security-VPC* and the **Subnet** to the *Public1* subnet. *Enable* **Auto-Assign Public IP** so we can access the instance through SSH. Press **Next: Add Storage**.

Do not change any storage settings. Press **Next: Add Tags**.

Give the *Name* tag a **Value** of *Bastion*. **Next: Configure Security Group**.

Create a new security group with a **name** of *Bastion*. Leave it so SSH is available. We are not using this as a web server, so no additional rules need to be added. **Review and Launch**.

Review the settings and press **Launch**. Set the key pair to our *security* key and click **Launch Instances**.

Wait for the hosts to finish provisioning.

Access the Instances

Copy the public IP of the Bastion host and open your workstation terminal. Navigate to the directory containing the sercurity.pem key pair file. We need to adjust the permissions of that file:

```
[user@workstation] chmod 400 security.pem
```

Add the key using ssh-agent:

```
[user@workstation] ssh-add -K security.pem
Identity added: security.pem (security.pem)
```

SSH into the server:

```
[user@workstation] ssh -A ec2-user@<BASTIONPUBLICIP>
```

We can now SSH into one of our private instances. Retrieve the private IP of one of the instances from the EC2 Dashboard. You may want to take this moment to rename the private instances something like *web1* and *web2* to differentiate them.

```
[ec2-user@bastion] ssh ec2-user@WEB1PRIVATEIP
```

However, what happens if we try to install Apache?:

```
[ec2-user@web1] sudo yum install httpd
```

The command times out, since our private servers have no route to the internet. To fix this, we need to add a NAT gateway to our setup.

NAT Gateway

From the **VPC Dashboard**, select **NAT Gateways** from the left menu. Press **Create NAT Gateway**. Set **Subnet** to *Public2*, then press **Create New EIP** to generate an elastic IP address. **Create a NAT Gateway**.

We now need to add our NAT gateway to our private subnets' route tables. Click **Edit Route Tables** and select the *Security-PrivateRT*. Press the **Routes** tab, then click **Edit**, **Add another route**. Set the **Destination** to *0.0.0.0/0* with a **Target** of the NAT gateway. **Save**.

Return to the **NAT Gateway** page and wait until the gateway's status reads as "available."

Back on the terminal, run the Apache install command again:

```
[ec2-user@web1] sudo yum install httpd
```

The install runs successfully. Start the Apache service, then exit out of the web1 server and log in to the web2 server:

```
[ec2-user@web1] sudo service httpd start
[ec2-user@web1] exit
[ec2-user@bastion] ssh ec2-user@WEB2PRIVATEIP
```

Install and start Apache:

```
[ec2-user@web2] sudo yum install httpd
[ec2-user@web2] sudo service httpd start
```

Elastic Load Balancer

Finally, we want to create an Elastic Load Balancer to serve traffic to our two web servers.

From the **EC2 Dashboard**, select **Load Balancers** from the left menu. Click **Create Load Balancer**, choose *Classic Load Balancer* and press **Continue**.

Set the **Load Balancer name** to *Security-ELB* and **Create [the] LB inside** our *Security-VPC*. Leave the **Listener Configuration** set to HTTP. We now need to select our subnets; however, ELBs can only use public subnets and our web instances are private. We can get around this as long as we have public subnetting in the same Availability Zone as our private subnets. Add both the *Public1* and *Public2* subnets. Press **Next: Assign Security Groups**.

We want to *Create a new security group* with a **Type** of *HTTP* so the **Port Range** of *80* is open. Press **Next: Configure Security Groups** then **Next: Configure Health Check**.

Set the **Ping Protocol** to *TCP* with a **Ping Port** of *80*. Change the **Healthy Threshold** to *2*. **Next: Add EC2 Instances**.

As we can see, both our *web1* and *web2* instances are available despite being outside of our public subnets -- this is because they are located in the same Availability Zones as our public subnets. Select both instances. Click **Next: Review and Create**.

Review the ELB's settings, then press **Create**. **Close** the resulting page to be directed back to the ELB dashboard and view the **Instances** tab. Wait for the instances to pass the health checks, then copy the DNS name from the **Descriptions** tab. Copy the URL into your browser to view the Apache test page.

This lab is now complete!