



Linux Academy

Hands-On Training

Creating a NAT
Instance in a VPC

Contents

Introduction.....	1
Scenario.....	1
Getting Started.....	1
Creating a Security Group.....	2
Adding Inbound Rules.....	2
Adding Outbound Rules.....	3
Create a NAT Instance.....	3
Allocating an IP.....	3
Note.....	3
Updating the Route Tables.....	5
Testing the NAT Instance.....	5

Introduction

In this lab, we will be applying concepts learned in the **AWS Certified Solutions Architect, Associate Level** course. More specifically, we will be addressing concepts from the VPC section of the course. We recommend you watch these videos if you have not already.

Scenario

We are creating a new private server that needs access only to our network, and is not publicly available; however, this instance will need to download software and updates from the Internet. To do this, we will need to create a NAT instance in our VPC to allow for updates and downloading needed software.

Getting Started

On the Linux Academy LiveLab page, log into the provided link with the provided credentials. From the AWS console, click on **VPC**, under **Networking**.

Before we go farther into configuring the NAT, we should first review the VPC that is already set up on the account. From the **VPC dashboard**, select **Subnets**. Two should be available with the names *Public* and *Private*. If you click on each Subnet, individually, you will see the *Public* subnet CIDR is set for *10.0.0.0/24*, while the *Private* is *10.0.1.0/24*.

The screenshot shows the AWS VPC console interface. On the left sidebar, 'Subnets' is selected under 'Your VPCs'. The main area displays a table of subnets:

Name	Subnet ID	State	VPC	CIDR	Availability Zone
Private	subnet-13ab5e4b	available	vpc-d6a1c1b2 (10.0.0.0/16)	10.0.1.0/24	us-east-1b
Public	subnet-152af963	available	vpc-d6a1c1b2 (10.0.0.0/16)	10.0.0.0/24	us-east-1b

Below the table, the details for the selected 'Private' subnet are shown:

- Subnet ID: subnet-13ab5e4b
- Name: Private
- CIDR: 10.0.1.0/24** (highlighted with a red box)
- State: available
- VPC: vpc-d6a1c1b2 (10.0.0.0/16)
- Availability Zone: us-east-1b
- Route table: rtb-4b0f552f
- Network ACL: acl-35fb8e51
- Default subnet: no
- Auto-assign Public IP: no
- Available IPs: 250

Having reviewed our subnets, move to the **EC2 dashboard** (located under **Services » Compute**). Here you will have two instances available for use. If you select an instance and view the description below, you can see which is associated with the Public or Private subnet, based on the *Private IPs*. The Public instance will have a *0* in the third number while the Private instance will contain a *1* in that same spot.

To ensure that our Private instance is really private, we need to open our terminal and SSH into our Public instance. Your username and password are provided on the Linux Academy LiveLab page; Server 1 will

be your Public server, 2 your Private.

After successfully getting into your Public server, SSH into your Private server from there — you will have to use the provided private IP.

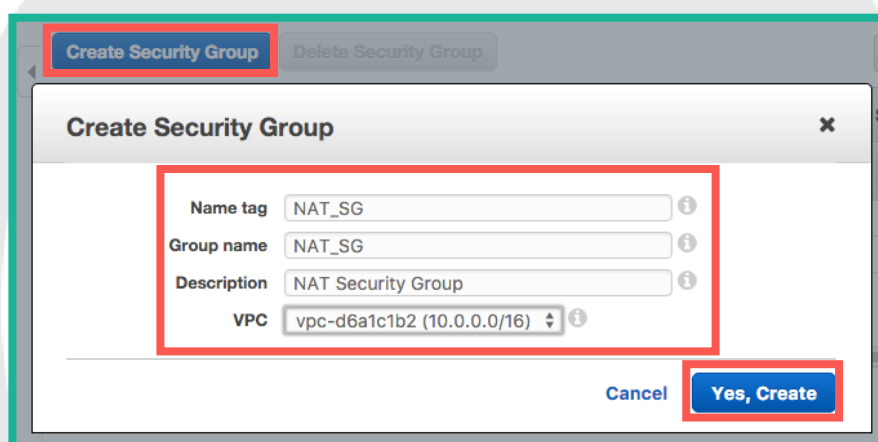
Next, try to download updates to your private instance:

```
sudo apt-get update
```

Your instance will hover at 0% because it is unable to connect to Ubuntu's mirrors since they are outside of our private network.

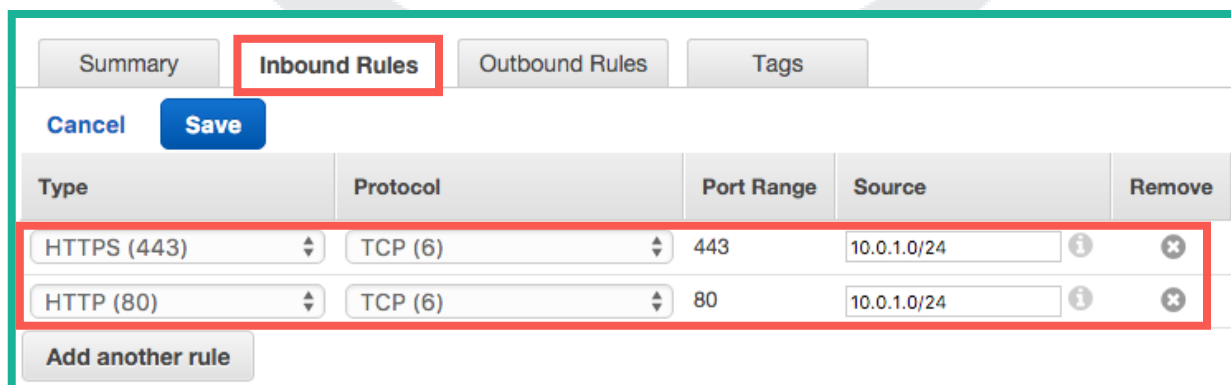
Creating a Security Group

Return to the **VPC dashboard**, and select **Security Groups**. Press **Create Security Group**. We have chosen to name ours *NAT_SG*. Select the **VPC** that includes your CIDR block (*10.0.0.0*). **Yes, Create**.



Adding Inbound Rules

With the *NAT_SG* security group selected, go to the **Inbound Rules** tab below. Press **Edit**. You will need to add rules for *HTTP (80)* and *HTTPS (443)*. Select these from the dropdown menu under **Type**. For **Source**, type in the name of your Private subnet's CIDR Block: *10.0.1.0/24*. **Save**.



Type	Protocol	Port Range	Source	Remove
HTTPS (443)	TCP (6)	443	10.0.1.0/24	✕
HTTP (80)	TCP (6)	80	10.0.1.0/24	✕

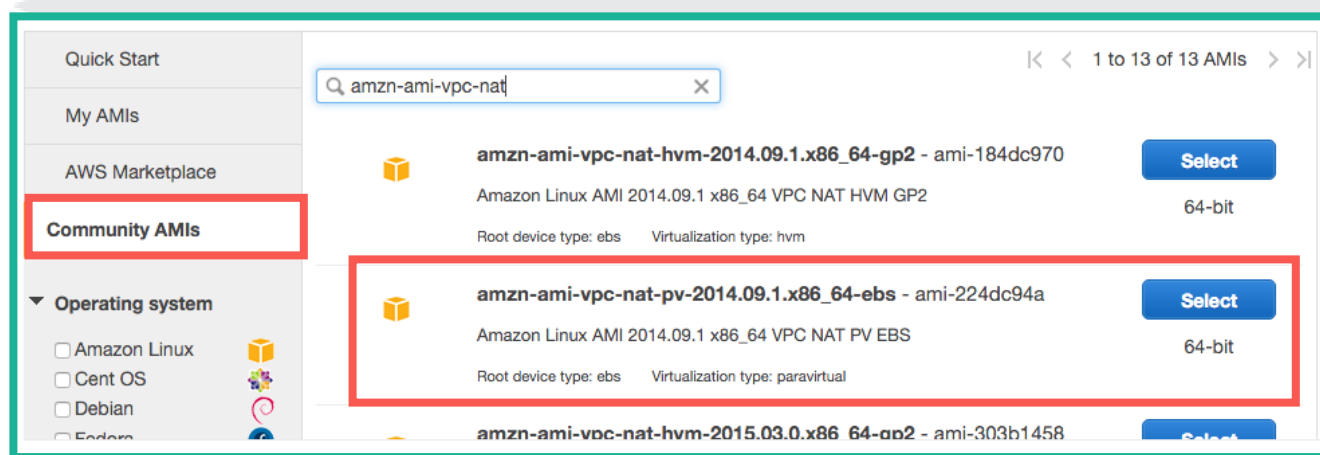
Adding Outbound Rules

By default, outbound rules will allow for *ALL Traffic*; this option is appropriate for our NAT instance, and can be left as is.

Create a NAT Instance

From your **EC2** dashboard, **launch a new instance**. The AWS community provides many already-created instances that can be used for a variety of specific tasks. In this case, we will be using the community's NAT image.

From the **Choose an Amazon Machine Image (AMI)** screen, select **Community AMIs**. Search for *amzn-ami-vpc-nat*. You will want to select the image titled *amzn-ami-vpc-nat-pv-2014.09.1.x86_64-ebs*.



For storage, select the *t1.micro*, then press **Next: Configure Instance Type**. Here, we want to select our VPC for the **Network**, and your *Public* subnet for the **Subnet**. Click **Next: Add Storage**.

The default storage settings are acceptable. Move on to **Tag Instance** — we will be giving our instance a **Value** of *NAT*. **Next: Configure Security Group**. **Select an existing security group**. Choose the group you just made in the steps above. **Review and Launch** your instance.

Note

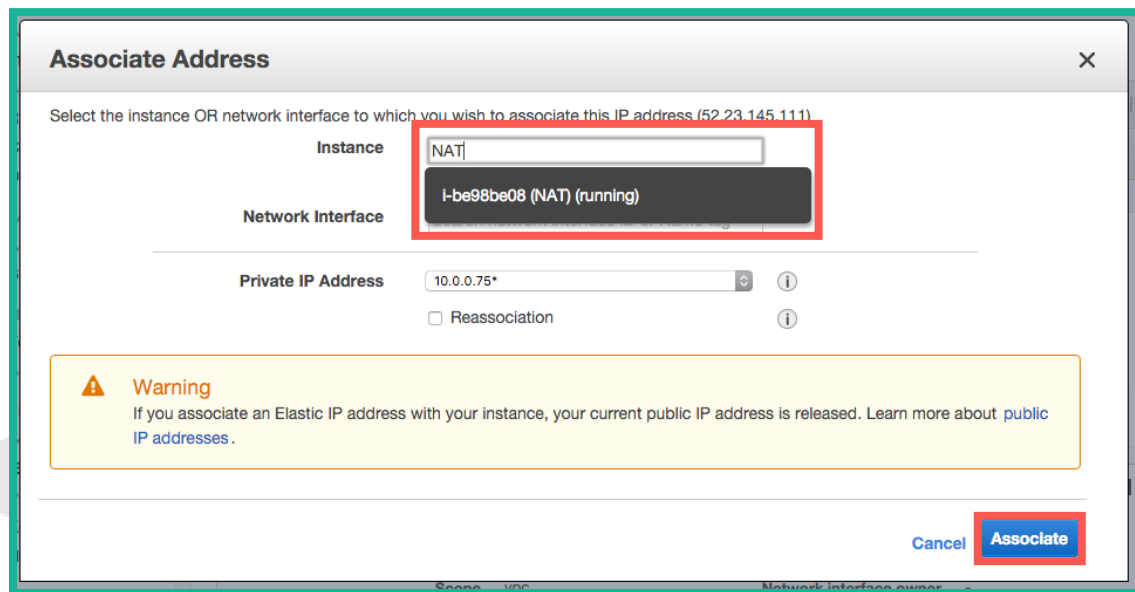
AWS will warn you with a pop-up that port 22 of the server is unopened. This is correct, and the warning should be ignored.

Launch your instance, and **Proceed without a key pair**.

Allocating an IP

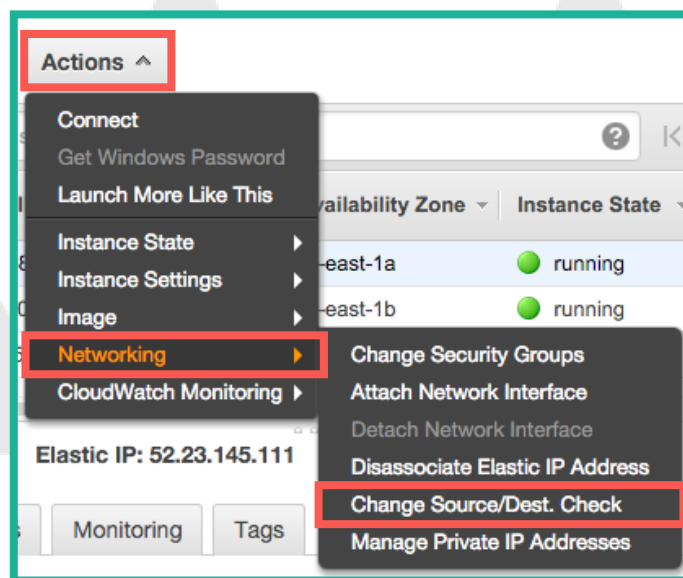
As our NAT instance launches, we need to create and assign an Elastic IP to the instance. From your **EC2 dashboard**, to go **Elastic IPs** on the left menu. Press **Allocate New Address**, then **Yes, Allocate**.

You will be presented with your new Elastic IP. Close the pop-up, then select the IP. From the **Actions** dropdown, click **Associate Address**, and search for your NAT instance. Press **Associate**.



We will now need to change the Source/Destination checks for our NAT instance. Return to the **Instances** screen, under the **EC2 dashboard**.

Select your NAT instance, and then under **Actions** » **Networking**, choose **Change Source/Dest. Check**. Click **Yes, Disable**.



With the NAT instance successfully set up, we need to alter our route tables. Before we do this, however, make note of your *NAT's instance ID*. You will need this later.

Updating the Route Tables

From the **VPC dashboard**, select **Route Tables**. You will notice one of the Route Tables is already associated with your *Public* subnet (the **Target** will be an Instance ID). Select the unassociated subnet, press the **Routes** tab below, and click **Edit**.

For **Target**, you will want to add your *NAT instance's ID*. For **Destination**, add *0.0.0.0/0*, and **Save**.

The screenshot shows the AWS VPC console interface for editing a route table. The 'Routes' tab is selected, and a new route is being added. The route table is named 'rtb-4b0f552f' and is associated with the VPC 'vpc-d6a1c1b2 (10.0.0.0/16)'. The 'Routes' tab shows a table with columns: Destination, Target, Status, Propagated, and Remove. A new route is being added with Destination '0.0.0.0/0' and Target 'i-be98be08'. The status is 'No' and it is not propagated. The 'Add another route' button is visible at the bottom.

Destination	Target	Status	Propagated	Remove
10.0.0.0/16	local	Active	No	
0.0.0.0/0	i-be98be08	No	No	

This allows the routing traffic from our private subnet to go to our NAT instance to download needed updates and software from the internet.

Testing the NAT Instance

Open your terminal, and using the same credentials as before, SSH into your Public instance, and then your Private one. From here, you can run:

```
sudo apt-get update
```

Although you are on a private network, the updates should run. If so, you have successfully created a NAT instance!