Linux Academy

Hands-On Training

Configuring VPC DNS

Contents

Introduction

In this lab, we will be configuring internal DNS using Route 53, as well as learning how to configure our VPC to use a different, on-premise DNS server, as though we were a company that has expanded to use AWS.

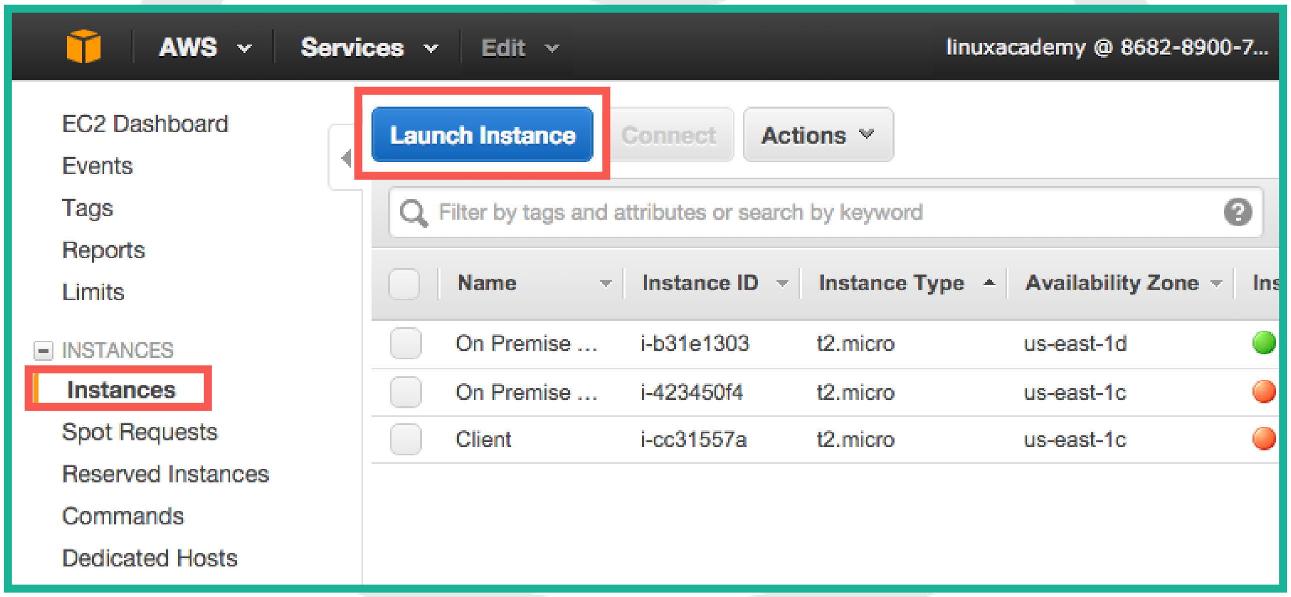
Getting Set Up

On the **LiveLab** page, select **Start Lab** and wait for the lab to generate. Once done, you will be provided with a link to the AWS Console and login information. Click on the link and log in with the given credentials.

Launching an EC2 Instance

We will be working with **Amazon EC2** as our virtual private cloud (VPC) and **Route 53** for DNS. An **EC2** instance is already provided under the account. If you select **EC2**, and then **Instances**, you can see the instance titled *On Premise DNS*.

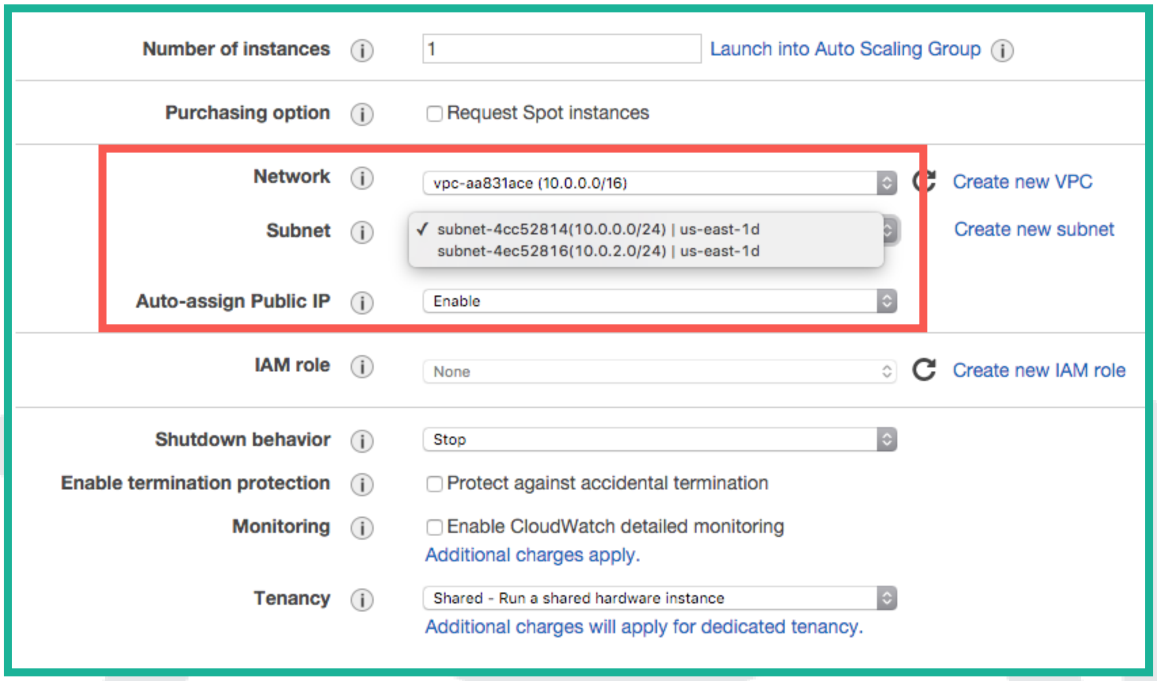
However, we will need an additional instance: Select **Launch Instance**.



Name	Instance ID	Instance Type	Availability Zone	Ins
On Premise ...	i-b31e1303	t2.micro	us-east-1d	●
On Premise ...	i-423450f4	t2.micro	us-east-1c	●
Client	i-cc31557a	t2.micro	us-east-1c	●

From here, we will select *Amazon Linux AMI*, then *t2.micro* as the **Instance Type**. Press **Next: Configure Instance Details**.

Ensure that the **Network address** is set to *(10.0.0.0/16)*, and the **Subnets available** are *(10.0.0.0/24)* and *(10.0.2.0/24)*. Select *Enable* for the **Auto-assign Public IP** option. Press **Next: Add Storage**, then **Next: Tag Instance**. We do not need to adjust the storage settings.



Number of instances	1
Purchasing option	☐ Request Spot instances
Network	vpc-aa831ace (10.0.0.0/16)
Subnet	✓ subnet-4cc52814(10.0.0.0/24) | us-east-1d subnet-4ec52816(10.0.2.0/24) | us-east-1d
Auto-assign Public IP	Enable
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.

In **Tag Instance**, use *Client* as the **Value**, and then press **Next: Configure Security Group**. Choose the **Select an existing security group** radio button, and select the one that contains *your* Linux Academy username. Press **View and Launch**.

From here, review your selections if needed, then select **Launch**. A pop-up box will ask for a key pair. Choose **Create a new key pair** from the drop-down list. We choose to name ours *dnslab*. Download the key pair, then select **Launch Instances**.

To ensure your instance works, open up your terminal, and navigate to the folder that contains your key pair download. (In this instance, our Downloads folder — this may vary depending on your computer settings.)

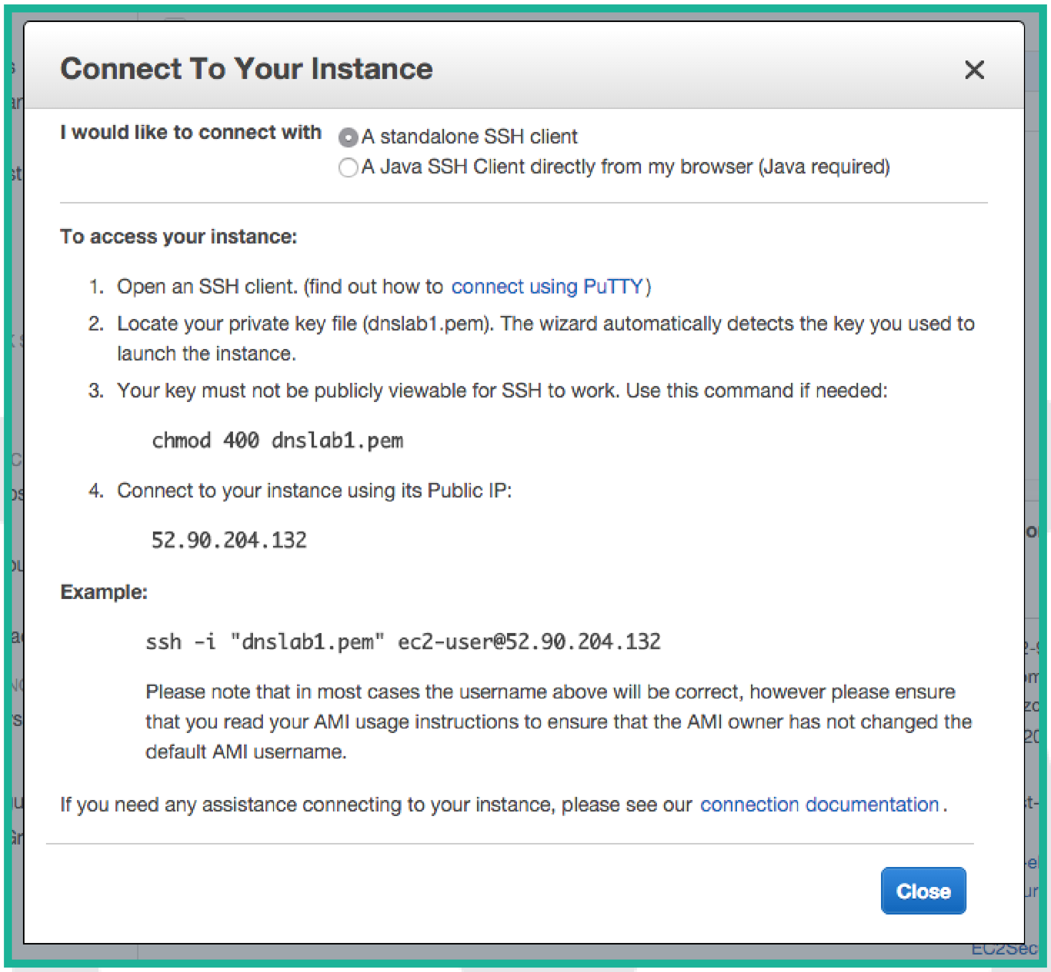
```
cd ~/Downloads
```

(Optional) Run `ls` to ensure the *dnslab.pem* file is available.

Change the permissions on the file:

```
chmod 400 dnslab.pem
```

Back in the AWS console, select the **Client** instance and select **Connect**, and connect with the ssh example provided (yours will differ from the example).



Connect To Your Instance

I would like to connect with [✓ A standalone SSH client] A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (dnslab1.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 dnslab1.pem
```

4. Connect to your instance using its Public IP:

```
52.90.204.132
```

Example:

```
ssh -i "dnslab1.pem" ec2-user@52.90.204.132
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

If you can ssh into your server, you're good to go! Be aware it may take a couple of minutes for your server to be ready.

Note

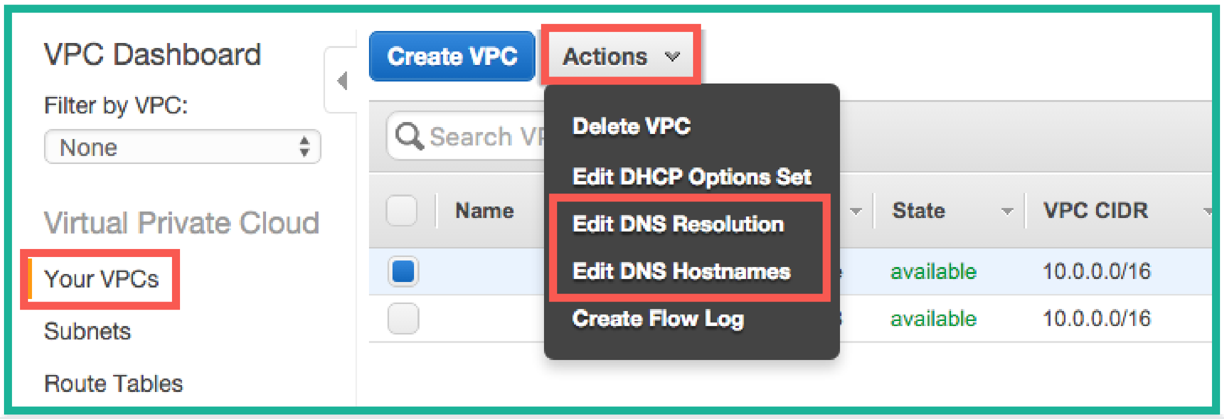
It may take time for the DNS to update. If the above steps fail, wait five minutes, then try again.

Configuring Internal DNS

Check Your VPC Settings

From the top **Services** menu, navigate to **Networking**, and then **VPC (Virtual Private Cloud)**. Select Your VPC from the menu.

At this point, we need to confirm that our VPC is properly configured to work with DNS — this means that DNS and hostname both need to be enabled. Check the available VPC, and go to **Actions**, **Edit DNS Resolution**. Ensure is it set to *Yes*. Do the same for **Edit DNS Hostnames**.

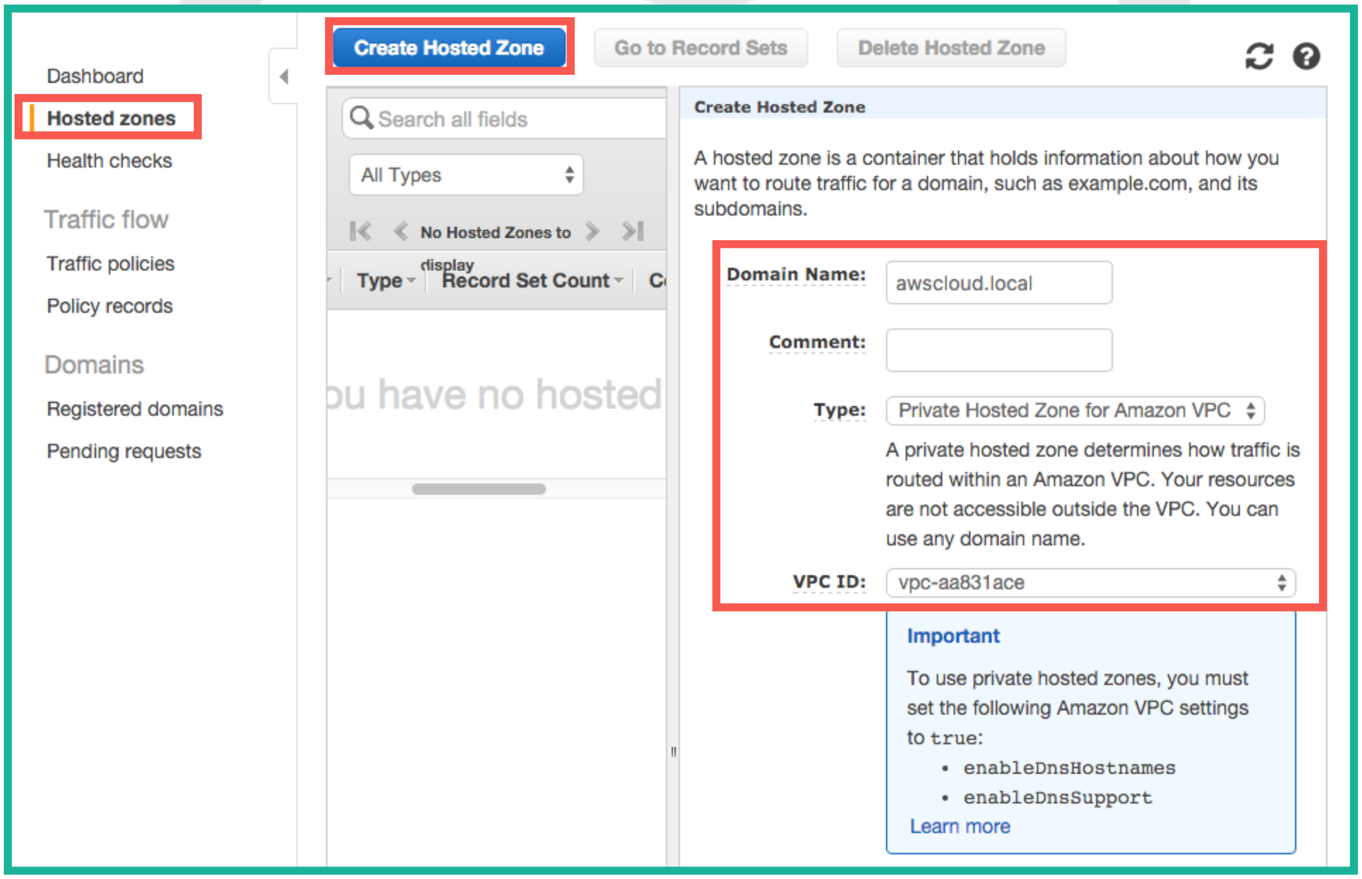


Name	State	VPC CIDR
	available	10.0.0.0/16
	available	10.0.0.0/16

With the proper setting confirmed, we can now set up our DNS.

Set Up DNS

Navigate to **Route 53**, also located under *Networking*. You may receive a permissions error, but this can be ignored. From there, move to **Hosted zones**, and select **Create Hosted Zone**.



Create Hosted Zone

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain Name:	awscloud.local
Comment:	
Type:	Private Hosted Zone for Amazon VPC
	A private hosted zone determines how traffic is routed within an Amazon VPC. Your resources are not accessible outside the VPC. You can use any domain name.
VPC ID:	vpc-aa831ace

Important

To use private hosted zones, you must set the following Amazon VPC settings to `true`:

- `enableDnsHostnames`
- `enableDnsSupport`

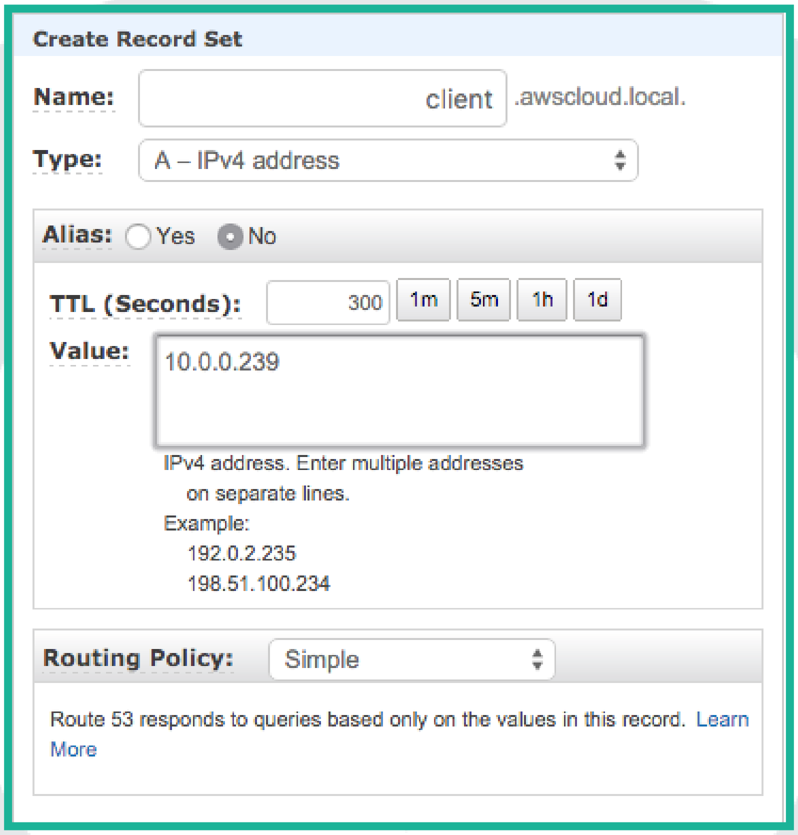
Learn more

In actual practice, the domain named used would be related to your organization. In this lab, we will be using the domain *awscloud.local* — the local implies it is an internal zone. Set *Private Hosted Zone for Amazon VPC* for **Type**, and select the VPC in the *US East (N. Virginia)* region for **VPC ID**. To confirm this is correct, check the VPC name against the VPC ID in your VPC Dashboard. Select **Create**.

Create an Internal Record Set

From the dashboard of your hosted zone, select **Create Record Set**.

Set the **Name** to *client*. Next, set the **Value** to the *private IP of your Client instance*, which can be found under the **EC2 dashboard**, by clicking on **Instances** and selecting your **Client** instance. Select **Create**.



Create Record Set

Name: client .awscloud.local.

Type: A – IPv4 address

Alias: Yes [✓ No]

TTL (Seconds): 300 1m 5m 1h 1d

Value: 10.0.0.239

IPv4 address. Enter multiple addresses on separate lines.
Example:
192.0.2.235
198.51.100.234

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. Learn More

Now, switch to your terminal. Ensure you are logged into your *Client* instance. Run:

```
nslookup client.awscloud.local
```

It should output results similar to:

```
Server:       10.0.0.2
Address:    10.0.0.2#53

Non-authoritative answer:
Name:    client.awscloud.local
Address: 10.0.0.72
```

The IP in the **Address** field should be your *Client instance's private IP* that was added in your **Route 53 client.awscloud.local** rules.

Now type:

```
sudo cat /etc/resolv.conf
```

This will output your **search** setting, and the **nameserver** will be the internal DNS server. If secondary DNS records were added, it would failover to the second, should AWS's DNS cease to work.

Configuring On-Premise DNS

Note

This section will not cover building a Linux DNS server. For more information on how to build a DNS server see our Linux courses. For this lab, the instance titled On Premise DNS will be used.

We want to allow our AWS applications to reference servers and instances in the "On Premise" DNS servers. However, this cannot be done with Route 53, since those must originate within the AWS cloud.

Connect to your *Client* from your terminal, and view `/etc/resolv.conf`:

```
sudo cat /etc/resolv.conf
```

From here we can see that we're still looking at Amazon's internal DNS. Any secondary servers added will work as a failover, so if we wish to use our on-premise internal DNS, we need to replace our internal AWS DNS.

Open `/etc/resvolv.conf`:

```
sudo vi /etc/resolv.conf
```

Now change the nameserver to match the on-premise private IP, which is *10.0.2.100*. Save and exit.

There is a DNS record located "on premise" that we have available. To check that everything works, run:

```
nslookup ns1.onpremise.local
```

You should get a resolved return:

```
Server:     10.0.2.100
Address:   10.0.2.100#53
```

```
Name:    ns1.onpremise.local
Address: 10.0.2.100
```

Should you `ping ns1.onpremise.local`, you will notice that it is returning the address. However, should you `ping ns1` itself, it will not work, because, by default, it will assuming you are trying to run `ping ns1.ec2.local`. This can be changed by altering the `/etc/resolv.conf` file:

```
sudo vi /etc/resolv.conf
```

We need to switch the **search** value to read *onpremise.local*:

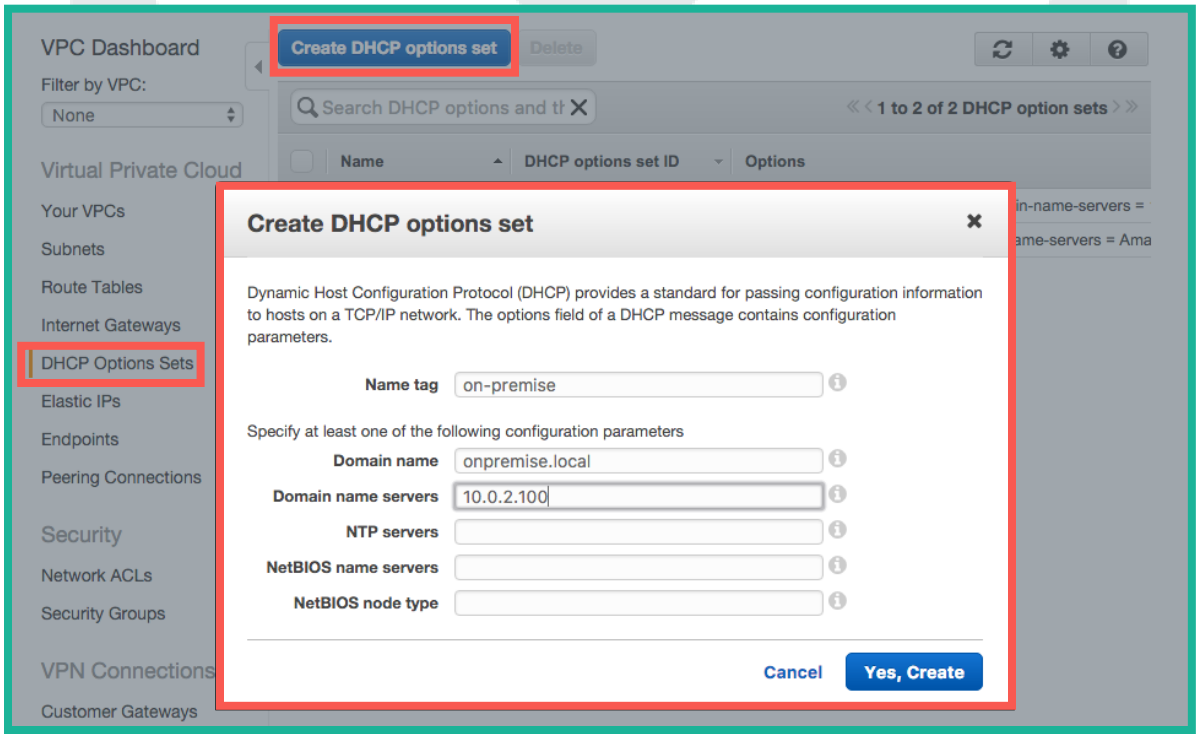
```
; generated by /sbin/dhclient-script
search onpremise.local

nameserver 10.0.2.100
```

Should you now `ping ns1`, it will work.

Set All Instances to Use On Premise DNS

If you are in a circumstance where you do not wish to use Route 53 at all, you can do so. Navigate to your **VPC dashboard**, and select **DHCP Options Set**. From there, click **Create DHCP options set**.

Set the **Name tag** to *on-premise*, and the **Domain name** to *onpremise.local*. For this example, the **Domain name server** should be set to our On Premise server's private IP: *10.0.2.100*. Press **Yes, Create**.



Create DHCP options set

Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The options field of a DHCP message contains configuration parameters.

Name tag	on-premise

Specify at least one of the following configuration parameters

Domain name	onpremise.local
Domain name servers	10.0.2.100
NTP servers	
NetBIOS name servers	
NetBIOS node type	

From the sidebar, choose **Your VPCs**. Select your VPC, and select **Edit DHCP Options Set** from the **Actions** menu. Choose the one with the *on-premise* tag, and **save**.

The changes have yet to take effect, and will not do so automatically. For them to do so, either type `sudo reboot` in your terminal or do it through your EC2 console. When the instance is newly-booted, return to your terminal and ensure you can `ping ns1`.

Conclusion

In this lab, we have learned how to use internal Route 53 DNS, which can only be used with the AWS private cloud. Because of this, we also learned how to use an on-premise DNS server, and how to get it running on your cloud clients.